Universidade de Aveiro
2010

Departamento de Electrónica
Telecomunicações e Informática

Tiago Amorim Ribeiro Gomes Pereira

Implementação de Tx/Rx banda base para 802.11-2007 em FPGA

dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Electrónica e Telecomunicações, realizada sob a orientação científica do Professor Manuel Alberto Reis de Oliveira Violas do Departamento de Electrónica Telecomunicações e Informática da Universidade de Aveiro

Aos meus pais e irmão

o júri

Presidente

Prof. Atílio Manuel da Silva Gameiro
Professor Associado da Universidade de Aveiro

Prof. Manuel Alberto Reis de Oliveira Violas
Professor Auxiliar da Universidade de Aveiro (Orientador)

Prof. Carlos Miguel Nogueira Gaspar Ribeiro
Professor Adjunto da Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Leiria (Arguente Principal)

agradecimentos

À minha família que me ajudou ao longo do percurso de toda a minha educação, mas principalmente ao meu pai, à minha mãe e ao meu irmão. Também ao meu tio que me deu uma grande ajuda na escrita desta tese e apoio durante a minha educação.

A todos os professores e alunos do Departamento de Electrónica Telecomunicações e Informática e do Instituto de Telecomunicações que contribuíram não só para a execução deste trabalho de alguma maneira, mas também no meu crescimento pessoal. Nomeadamente ao Professor Manuel Violas pela orientação que me ofereceu durante este trabalho, mas principalmente por estar sempre disposto a tirar-me qualquer dúvida que me surgiu no seu desenvolvimento e/ou para me apontar para as pessoas correctas que soubessem resolver os problemas que encontrei ao longo deste trabalho.

Ao Diogo Louro, o meu colega de casa de momento, mas também a todos os outros que moraram comigo ao longo deste percurso. Temos momentos inesquecíveis juntos.

Ao Rui, Serrano, Xico e JT; por todos os fins de semana, férias ou qualquer momento em que estamos juntos. São dos meus melhores amigos.

Ao Luís Ribeiro e David Campos. Nem conseguiria começar a explicar porquê.

Ao Hugo Carvalho e Andreia Migueis, é só rir quando se está com vocês.

Por fim, ao Filipe Rodrigues. Porque é sempre preciso um professor e um amigo quando isto aperta.

palavras-chave

OFDM, Xilinx System Generator, Field-Programmable Gate Array, 802.11, Hardware Co-Simulation

resumo

O trabalho apresentado nesta dissertação teve como objectivo o desenvolvimento da camada física de um sistema de transmissão e recepção de sinais OFDM baseados no standard IEEE 802.11-2007. O sistema desenvolvido inclui geração de dados aleatórios, modulador QAM, inserção de pilotos e subportadora DC, IFFT com adição de Prefixo Cíclico, buffer de saída e o consequente oposto para o receptor.

A dissertação encontra-se dividida em duas partes principais. Na primeira parte, o sistema foi projectado e simulado em Matlab através do ambiente Simulink com o auxílio dos blocos da Xilinx inseridos no seu software System Generator for DSP. Na segunda parte, foram adicionadas DAC's ao transmissor e o próprio foi compilado para um bloco e testado no XtremeDSP Development Kit-IV da Nallatech que inclui uma Field-Programmable Gate Array.

Todos os módulos foram desenhados usando os blocos do System Generator for DSP da Xilinx. O kit está conectado ao computador através de uma interface PCI. Os dados obtidos são exibidos em Matlab para a primeira parte e num osciloscópio para a segunda parte.

keywords

OFDM, Xilinx System Generator, Field-Programmable Gate Array, 802.11, Hardware Co-Simulation

abstract

It was the objective of this dissertation the development of the Physical Layer of an IEEE 802.11-2007 Transmitter-Receiver system for generating OFDM signals. The developed design includes random Data Generation, QAM Modulator, Pilots and DC subcarrier insertion, IFFT with Cyclic Prefix insertion, an Output Buffer and the subsequent opposite for its receiver.

This dissertation was divided in two main segments. In the first segment, the system was designed and simulated in Matlab through the Simulink environment using Xilinx's System Generator for DSP blocks. In the second part, DAC's where added to the transmitter in order to compile it into a single block and test it on Nallatech's XtremeDSP Development Kit-IV, which includes a Field-Programmable Gate Array.

All modules were designed using Xilinx's System Generator for DSP blocks. The kit is connected to the computer through a PCI interface. Output data is displayed on the Matlab environment for part one and on an oscilloscope for part two.

Table of Contents

List of Figures

List of Tables

Acronyms

ACK	*Acknowledgement*
ADC	*Analog-to-Digital Converter*
ADSL	*Asymmetric Digital Subscriber Lines*
ASIC	*Application-Specific Integrated Circuit*
AWGN	*Additive White Gaussian Noise*
A-MPDU	*MAC protocol data unit aggregation*
A-MSDU	*MAC service data unit aggregation*
BACK	*Block Acknowledgement*
BCC	*Binary Convolutional Coding*
BPSK	*Binary Phase-Shift Keying*
CAD	*Computer-Aided Design*
CCK	*Complementary Code Keying*
CDMA	*Code Division Multiple Access*
CLB	*Configurable Logic Blocks*
CP	*Cyclic Prefix*
CPLD	*Complex Programmable Logic Device*
CSMA/CA	*Carrier Sense Multiple Access with Collision Avoidance*
DAB	*Digital Audio Broadcast*
DAC	*Digital-to-Analog Converter*
DDC	*Digital Down Converter*
DMT	*Discrete Multi Tone*
DSP	*Digital Signal Processing*
DSSS	*Direct-Sequence Spread Spectrum*
DVB-T	*Digital Video Broadcast-Terrestrial*
FDM	*Frequency Division Multiplexing*
FDMA	*Frequency Division Multiple Access*
FEC	*Forward Error Correction*
FFT	*Fast Fourier Transform*
FHSS	*Frequency-Hopping Spread Spectrum*
FMCW	*Frequency Modulated Continuous Wave*

FPGA	*Field-Programmable Gate Array*
HDL	*Hardware Description Language*
HDSL	*High bit-rate Digital Subscriber Lines*
IC	*Integrated Circuit*
ICI	*Inter Channel Interference*
IF	*Intermediate Frequency*
IFFT	*Inverse Fast Fourier Transform*
IR	*Infrared*
ISI	*Inter Symbol Interference*
LAN	*Local Area Network*
LDPC	*Low-Density Parity-Check*
LFSR	*Linear Feedback Shift Register*
LUT	*Lookup Table*
MAC	*Multiply-Accumulate Operations per second or Medium Access Control*
MCS	*Modulation and Coding Schemes*
MPDU	*MAC Protocol Data Unit*
MSDU	*MAC Service Data Unit*
NRE	*Non Recurring Expense*
OFDM	*Orthogonal Frequency-Division Multiplexing*
OFDMA	*Orthogonal Frequency-Division Multiple Access*
PAPR	*Peak-to-Average Power Ratio*
PHY	*Physical Layer*
QAM	*Quadrature Amplitude Modulation*
QPSK	*Quadrature Phase-Shift Keying*
RAM	*Random Access Memory*
ROM	*Read-Only Memory*
SDR	*Software-Defined Radio*
SRAM	*Static Random Access Memory*
STBC	*Space-Time Block Coding*
SysGen	*Xilinx System Generator for DSP*
TxBF	*Transmit Beamforming*
UART	*Universal Asynchronous Receiver/Transmitter*
VHDL	*Very-High-Speed Integrated Circuit Hardware Description Language*
WLAN	*Wireless Local Area network*

Chapter 1

Introduction

1.A) Motivation

Software Defined Radio (SDR) has been an emerging concept within the wireless industry with great potential to be applied among a vast range of applications. Created by Joseph Mitola in 1991 [4], it can be defined as radio communication where some of the Physical Layer components that are typically implemented in analog hardware, such as filters, mixers, modulators, among others, are implemented in software instead with the aid of a Digital Signal Processor, thus allowing users to operate the radio in different environments providing the system with high flexibility. One main advantage of this technology is its reconfigurability given that changing its purpose requires changing its software, or in some cases upgrading it. The limitations imposed nowadays on SDR technology lie on the need for an analog front-end in order to upconvert or downconvert from a baseband signal to an *Intermediate Frequency* (IF) or vice-versa, respectively.

Our typical radio systems nowadays such as house or computer radio systems support at least four different radio standards (a/b/g/n) with dedicated circuits for filtering, modulating and synchronizing for each standard. Such radio would not be feasible in terms of size and power consumption. SDR can solve this problem given that a single radio can support several different standards, although it maintains its need for an analog front-end, which is typically common for any existing *Wireless Local Area network* (WLAN) standard.

System Generator for DSP is a blockset provided by Xilinx for designing and modeling FPGA-based DSP systems that is integrated into Matlab's Simulink, enabling hardware design by allowing the blocks to be synthesized into a FPGA, thus allowing the user to abstract himself from a time consuming and knowledge-dependent programming language such as *Very-High-Speed Integrated Circuit Hardware Description Language* (VHDL) or Verilog.

1.B) Related Work

Xilinx's System Generator for DSP has been playing an important role in the design and implementation of digital systems on FPGAs, especially in baseband processing for wireless radio. A designer can now model and implement an either simple or complex digital system while avoiding the complex learning of VHDL. Dejan Dramicanin et al. [5] built an IEEE 802.11a baseband processor Physical Layer prototype on a FPGA platform with main focus on the 802.11a synchronization preamble. The paper shows that FPGA designs can present a realistic method to equip and maintain high-tech laboratories, also with the possibility to drive even more advanced project practice. Dick Benson [6] shows the design of a simple **Single Sideband** (SSB) generator using Weaver's scheme with an audio input created from a weighted sum of sine waves. The algorithm's bitstream file generated from the map, place and route process typical to high-density PLD implementations is accomplished with a single mouse click. A basic user Interface is also created from switches and sliders for control and information purposes. On [7], Fey-yu et al. proposes a WCDMA *Digital Down Converter* (DDC) that includes a Direct Digital Synthesizer submodule produced with Xilinx DDS Compiler [9] and DSP48E slices [8]. Remaining submodules are designed with Xilinx FIR Compiler [10]. The DDC must satisfy the 3GPP specifications that define the transmission/reception requirements for a WCDMA's radio. The full system is composed of a DDS, mixer, half-band decimation and RRC filters. A similar work is present on [11], but for a WCDMA Digital Up Converter.

1.C) Programmable Logic Devices

Although Chang [1] initially developed OFDM in 1966, only in 1995 did it start being used for high-speed communications within the Digital Audio Broadcasting (DAB) standard.

Given that OFDM is carried out in the digital domain, there are a number of platforms to implement an OFDM system.

The most common method to design a system nowadays is through the use of *Field-Programmable Logic Devices* (PLD). PLD is a generic term referring to any kind of *Integrated Circuit* (IC) used for implementing hardware in the digital domain [2]. Below, three PLD platforms are described:

i) Application Specific Integrated Circuit (ASIC)

Application Specific Integrated Circuits (ASICs) (Figure 1.1: Application Specific Integrated Circuit (ASIC)) are the lowest form of PLDs and one of those possible methods. These are ICs built for a particular use and as such are also the smallest and lowest power usage method to implement an OFDM system in hardware. However, this method also comes with an increased manufacturing time, designing time and a higher skill requirement on the designing team's side.

Figure 1.1: Application Specific Integrated Circuit (ASIC)

ii) Complex Programmable Logic Device (CPLD)

Another method is the use of a *Complex Programmable Logic Device* (CPLD). These consist of an arrangement of multiple ASIC-like blocks offering a wide number of inputs and are often referred as having coarse-grained architecture. Presenting a lower of ratio §of flip-flops to logic resources, they can only be matched by FPGAs. Due to their simpler architecture, CPLDs are usually cheaper when compared to FPGAs and best suited for relatively small designs, therefore being used for simple logic applications. They have large logic building blocks and a centralized interconnection structure that gives them a fast and predictable performance.

Figure 1.2: Complex Programmable Logic Device (CPLD)

iii) Field-Programmable Gate Array (FPGA)

A FPGA is a reconfigurable logical device consisting of a “sea of NAND gates” and is characterized by a structure that allows a very high logic capacity. Unlike CPLDs, it offers more narrow logic resources. An FPGA consists of an array of small logic blocks and distributed interconnect resources, which gives it a slower pin to pin performance due to routing (pipelining can help though); both of these are the features of an FPGA, and are configured by the end-user programming. Due to its importance on this project, Chapter 4 is dedicated to the FPGA.

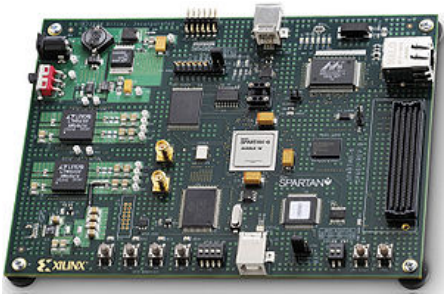


Figure 1.3: Field-Programmable Gate Array (FPGA)

This problem resolution is performed on chapter **4.C) ASIC, FPGA or CPLD?.**

1.D) Background and Contributions

This thesis is a follow-up from the previous master student project entitled "FPGA-based Vector Signal Generator" by Afonso Resende Silva [3]. On his project, he designed an OFDM Transmitter partially adapted to the IEEE 802.16 standard fully working in hardware. His work focuses mainly on QAM modulation, pilot insertion and IFFT.

He also gave me access to his Xilinx online account, Xilinx System Generator projects and VHDL files for clock processing along with some basic knowledge on understanding his system and the System Generator workspace.

1.E) Objective

The goal of this project is the development of an IEEE 802.11 Transmitter-Receiver system in order to transmit OFDM data symbols at a physical (PHY) layer level according to the OSI layer model (Figure 1.4). The transmitter side should include random Data Generation, 16-QAM Modulator, Pilot Tones and DC subcarrier insertion, *Inverse Fast Fourier Transform* (IFFT) with Cyclic Prefix Insertion and an Output Buffer. The receiver sector should contain an Input Buffer, Cyclic Prefix Removal, *Fast Fourier Transform* (FFT), OFDM De-Assembly Module (Pilots/DC Removal) and a 16-QAM Demodulator.

System development will be performed and divided into two main parts. On the first part, the whole system will be designed and simulated in Matlab on a Simulink environment using Xilinx's *System Generator for DSP* blocks. This part's receiver should be able to recover the signal created on the transmitter's side. In the second segment, the transmitter will be compiled into a library block and tested in the *XtremeDSP Development Kit-IV* using the *Hardware Co-Simulation* compilation mode in Simulink.

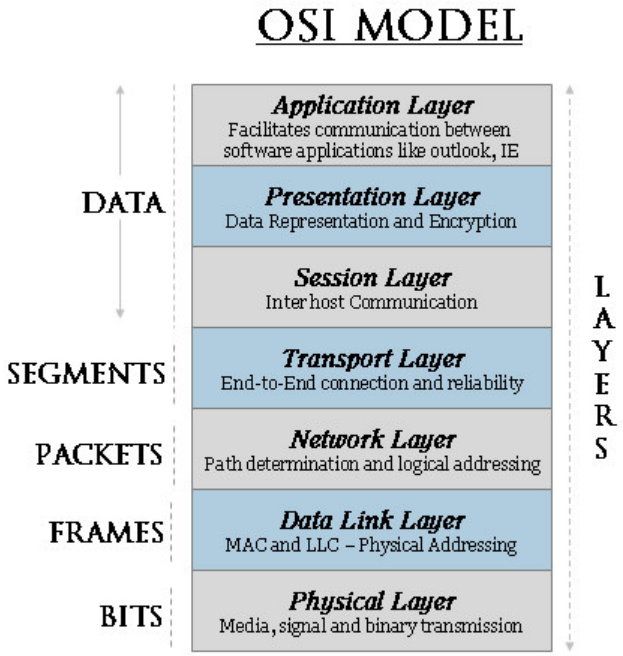


Figure 1.4: OSI Layer Model [12]

1.F) Outline

This project is presented throughout a total of seven chapters, this being the introductory one. The following three chapters present the preparation for the intended work, followed by two chapters related to the implementation that was made. The project ends with the reached conclusions and possible future work. A content description on each chapter is performed below:

- **Chapter 1: Introduction** presents why the interest on the thesis development, what problem is there to be solved, the objectives to be achieved and the main contributions on it.
- **Chapter 2: IEEE 802.11 group and amendments** offers an overview on the amendments brought by the IEEE 802.11 group mainly dedicated for Wi-Fi as well as brief description of how the latest amendment achieves 600 Mbps on the PHY layer.
- **Chapter 3: Orthogonal Frequency-Division Multiplexing** starts by making a description of a digital communication system followed by the story of how OFDM came to be with some of the main events along its creation. The chapter continues with a set of advantages, disadvantages and applications on OFDM and is ended by OFDM characteristics on 802.11-2007 with some of the characteristics relevant for the project.
- **Chapter 4: Field-Programmable Gate Array** shows a generic description of the platform used for the system design and why was it the chosen one among other platforms. An overview of main CAD environment used is also made. The chapter ends with a brief description of the most recent models offered by Xilinx for DSP purposes.
- **Chapter 5: Simulation of an OFDM Signal Transmitter and Receiver on Matlab** illustrates the design that was implemented on Matlab described in two subchapters, one for the transmitter and another for the receiver, with each subchapter making a detailed description on each main block.
- **Chapter 6: FPGA Co-Simulation of an OFDM Signal Transmitter** shows the implementation on FPGA of the transmitter design that was simulated on Matlab through means of Hardware Co-Simulation. The FPGA resources in use are also presented.
- **Chapter 7: Conclusions and Future Work** presents the conclusions obtained with this work and some possibilities for its continuation in the future.
- **Annex A: XtremeDSP Development Kit-IV** presents a description of the main aspects of the platform used for the thesis.
- **Annex B: Block Diagrams of the developed system** illustrates all the blocks that were used on the thesis.

Bibliography

[1] Chang, R. W., “Synthesis of band-limited orthogonal signals for multi-channel data transmission”, Bell System Technical Journal 46, pp. 1775-1796, 1996.

[2] S. Brown & J. Rose, “Architecture of FPGAs and CPLDs: A Tutorial”, IEEE Design and Test of Computers, Vol. 13, Issue 2, pp. 42-57, 1996.

[3] Afonso Resende Silva, “Geração de Sinais Vectoriais Baseada em FPGA", Masters Thesis, Universidade de Aveiro, 2008.

[4] Joseph Mitola, "The Software Radio," IEEE National Telesystems Conference, 1992.

[5] Dejan M. Dramicanin et al., “FPGA-based Prototyping of IEEE 802.11a Baseband Processor”, Serbian Journal of Electrical Engineering, Vol.1 No. 3, pp. 125-136, November 2004.

[6] Dick Benson and Narinder Hall, “System-Level Design Using FPGAs and DSPs: An Example Showing Software-Defined Radio”, RTC Magazine, January 2004.

[7] LIN Fei-yu et al., “Efficient WCDMA Digital Down Converter Design Using System Generator”, International Conference on Space Science and Communication, Malasya, October 2009.

[8] Xilinx Inc., “Virtex-5 FPGA XtremeDSP Design Considerations”, User Guide v3.4, June 2010.

[9] Xilinx Inc., “LogiCORE IP DDS Compiler v5.0”, September 2010.

[10] Xilinx Inc., “LogiCORE IP FIR Compiler v5.0”, April 2010.

[11] Wang Wei et al., “Efficient Wireless Digital Up Converters Design Using System Generator”, ICSP 2008, pp. 443-446, December 2008.

[12] http://www.networkguruz.com/wp-content/uploads/2008/06/osi-reference-model.jpg.

Chapter 2

IEEE 802.11 group and amendments

2.A) Introduction

Since 1999, 802.11 has been the tip of the movement in mobile data networking [1]. IEEE 802.11 is a group of amendments (most often referred to as standards), where most of them are dedicated to *Wireless Local Area Networks* (WLANs) along the 2.4 and 5 GHz frequency bands. 802.11-1997 (often referred to as IEEE 802.11 Legacy Mode) was the first wireless networking standard designed by the group and all the other standards created after are seen by the group as amendments to this one and not as new standards.

2.B) Standards

While there are more than 20 standards within the group, most of them are either obsolete or not as important for home WLANs as the ones mentioned below:

i) 802.11-1997 (Legacy Mode)

Created in June 1997 and nowadays obsolete, its operating frequency was on the 2.4 GHz band and was able to reach speeds of 1 (mandatory) or 2 (optional) Mbit/s along with *Forward Error Correction* (FEC) coding and *Carrier Sense Multiple Access with Collision Avoidance* (CSMA/CA). One disadvantage presented on this frequency band is the interference caused by other devices working on this range such as microwave ovens, PDA's and Bluetooth devices, among others. It possessed three alternative physical layer technologies: diffuse *infrared* (IR), *Direct-Sequence Spread Spectrum* (DSSS) or *Frequency-Hopping Spread Spectrum* (FHSS) and had an approximate indoor range of 20 meters.

	802.11	802.11a	802.11b	802.11g
Bandwidth (MHz)	300	83.5	83.5	83.5
Frequency Range (GHz)	2.4-2.4835	5.15-5.25 (lower) 5.25-5.35 (middle) 5.725-5.825 (upper)	2.4-2.4835	2.4-2.4835
Number of Channels	3	12 (4 per subband)	3	3
Modulation	BPSK,QPSK DSSS,FHSS	BPSK, QPSK, MQAM OFDM	BPSK,QPSK DSSS	BPSK, QPSK, MQAM OFDM
Coding		Conv. (rate 1/2,2/3,3/4)	Barker, CCK	Conv. (rate 1/2,2/3,3/4)
Max. Data Rate (Mbps)	1.2	54	11	54
Range (m)		27-30 (lower band)	75-100	30
Random Access	CSMA/CA			

Table 2.1: IEEE 802.11a/b/g/n Specifications [6]

ii) 802.11a

Born in September 1999 along with standard 802.11b, it has the same frame format as the Legacy Mode standard. Its encoding scheme, unlike 802.11b and the initial standard, is *Orthogonal Frequency-Division Multiplexing* (OFDM). Also differs from both standards by operating on the wider 5 GHz band; giving the standard a big advantage due to this frequency zone low use at the time it was created, allowing more signals to coexist without interference. Even so, the 5 GHz band is divided into 8 channels to lighten its use. However, standards operating at high frequencies also bring disadvantages, such as higher attenuation and lower range due to the signal's smaller wavelength.

Also, since it works on a different frequency band, it's not compatible with 802.11b products, thus requiring bridging products so both bands are supported. The standard can achieve several data rates ranging from 6 Mbit/s when using *Binary Phase-Shift Keying* (BPSK) with a FEC rate of 0.5, to 54 Mbit/s when using 64-*Quadrature Amplitude Modulation* (QAM) with a FEC rate of 0.75 depending on the type of modulation applied by OFDM.

iii) 802.11b

Ratified in 1999, was the first standard to be widely recognized. Most of its core is based on standard 802.11-1997, for example its multiple access method (CSMA/CA) and one of its modulation schemes (DSSS), but brings improvements to speed by providing velocities of 5.5 and 11 Mbit/s by using *Complementary Code Keying* (CCK) for modulation purposes; also works on the 2.4 GHz frequency band. It became widely accepted due to 802.11b devices' price reduction and the throughput boost it got when compared to 802.11-1997.

In order to prevent interference caused by user density and other devices on the 2.4 GHz band, 14 channels were defined along the band (2.4-2.4835 GHz); each with a 22 MHz width. There are 3 non-overlapping channels (1, 6 and 11) currently in use by the U.S.; Europe uses a different set (1, 5, 9 and 13). Although such technique attenuates interference, it doesn't erase it completely due to interference from side-lobes.

iv) 802.11g

Ratified in June 2003, works in the same frequency band as 802.11b (2.4 GHz) and uses the same modulation scheme that 802.11a (OFDM); also uses CCK or DSSS modulation, making its compatibility with 802.11b one of its biggest advantages. Can achieve the same throughputs that 802.11a, ranging from 6 to 54 Mbit/s with Adaptive Rate Selection just like the previous standards. Just like 802.11b, it suffers from the same interference issues; therefore it also divides the frequency band into several channels.

v) 802.11n

Ratified in October 2009, 802.11n is the latest amendment directed toward WLANs and can reach a PHY layer throughput of 600 Mbit/s [2][4] by adding *Multiple-Input Multiple-Output* (MIMO) antennas with diversity techniques (Figure 2.1) to the *Physical Layer* (PHY) and frame aggregation to the MAC layer; along with several spatial diversity strategies such as *Space-Time Block Coding* (STBC) and feedback methods like Implicit and Explicit *Transmit Beamforming* (TxBF), among other characteristics [2].

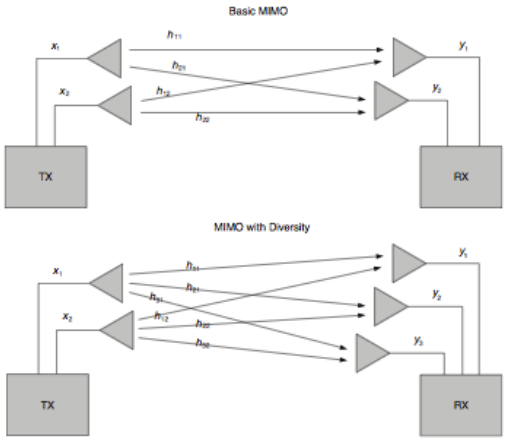


Figure 2.1: MIMO system with diversity receiver [2]

It presents three types of OFDM Preamble formats: Legacy, compatible with 802.11a/b/g devices; Mixed mode, for compatibility with both Legacy devices and high-throughput MIMO operation and Greenfield Mode, for high-throughput MIMO operation but not detectable by Legacy devices. The previous standards use 20 MHz bandwidth channels; 802.11n supports these channels and also supports optional 40 MHz channels.

Binary Convolutional Coding (BCC) is its channel encoding technique as well as *Low-Density Parity-Check* (LDPC) coding, although LDPC is optional.

On the MAC layer, the receiver doesn't use *acknowledgements* (ACK) for each *MAC Protocol Data Unit* (MPDU) that is sent by the transmitter as the previous standards do. Instead, it uses *Block Acknowledgements* (BACK) for each set of MPDUs, which significantly reduces overhead; and with frame aggregation, multiple MPDUs are assembled into a single PHY frame. This type of aggregation is called *MAC protocol data unit aggregation* (A-MPDU). Another type of aggregation also exists on the standard entitled *MAC service data unit aggregation* (A-MSDU). Residing at the top of the MAC layer, it aggregates several MSDUs into a single MPDU with each MSDU being attached to a subframe header containing destination address, source address and a length field. Both aggregations provide a trade-off between latency and throughput since both of them increase [2][5].

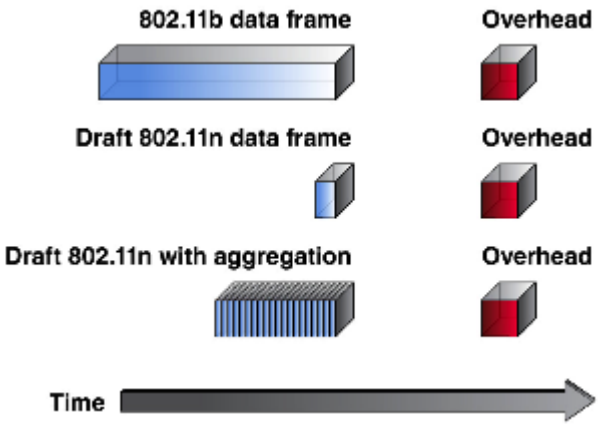


Figure 2.2: 802.11n data frame with aggregation [3]

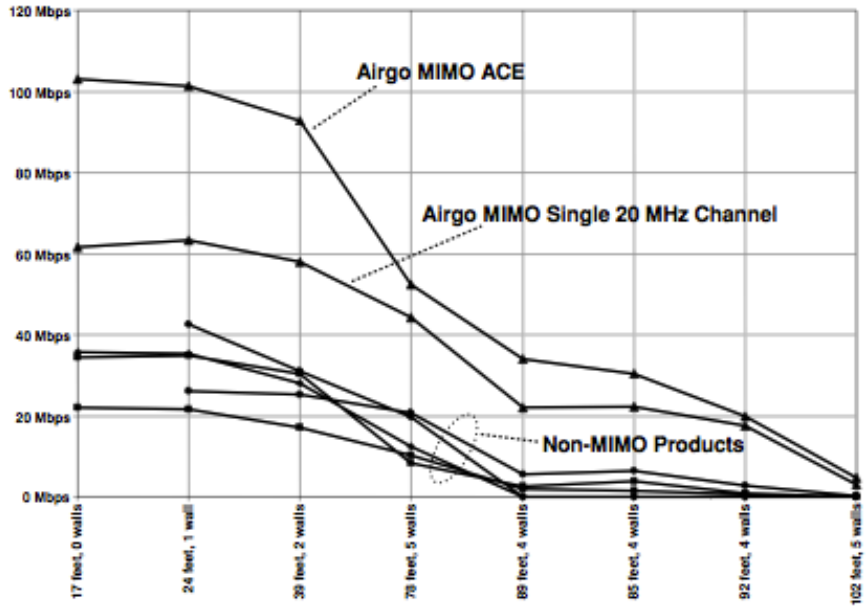


Figure 2.3: Throughput comparison for Non-MIMO and MIMO products [4]

The maximum number of possible simultaneous data streams provided by standard is limited by the minimum number of antennas on one side of the bond; hence, a link created by a transmitter with 2 antennas and a receiver with 3 antennas, can have either 1 or 2 data streams.

A total of 32 *Modulation and Coding Schemes* (MCS) exist according to the 3 parameters that define them: modulation type (BPSK, *Quadrature Phase-Shift Keying* (QPSK), 16-QAM or 64-QAM), number of spatial streams (1, 2, 3 or 4) and coding rate (0.5, 0.75, 5/6 or 2/3).

vi) How does 802.11n achieve 600 Mbit/s?

Even with the innovative MIMO-OFDM technology, going from 54 Mbit/s (802.11g) to 600 Mbit/s in raw bit rate may seem hard to understand. Starting with the maximum throughput of 802.11g, the techniques below show how the 600 Mbit/s are achieved:

a. Subcarriers boost

802.11g has 48 data subcarriers. 802.11n further increases the number of OFDM data subcarriers to 52, thus boosting throughput from 54 Mbit/s to 58.5 Mbit/s on a 20 MHz channel.

b. Higher FEC rate

802.11g's maximum FEC coding rate is 0.75. 802.11n further increases this coding rate by increasing it to 5/6, giving throughput another boost from 58.5 Mbit/s to 65 Mbit/s.

c. *Guard Interval* (GI) Clamp

Previous standards used a Guard Interval of 800ns. 802.11n brings an optional feature to reduce this GI to 400ns, thereby boosting throughput from 65 Mbit/s to 72.2 Mbit/s.

d. 40 MHz Channels

20 MHz channels were always a commonality in standards previous to 802.11n. However, the standard has an optional feature to extend its channel bandwidth from 20 to 40 MHz. With the channel extension the number of total OFDM data subcarriers gets more than doubled up, going from 52 to 108 subcarriers. This increase is shown on Figures 2.4 and 2.5; not forgetting that the 20 MHz channel has 4 pilot subcarriers and the 40 MHz channel has 6. Along with the improved FEC coding rate and the GI reduction, total throughput goes from 72.2 Mbit/s to 150 Mbit/s.

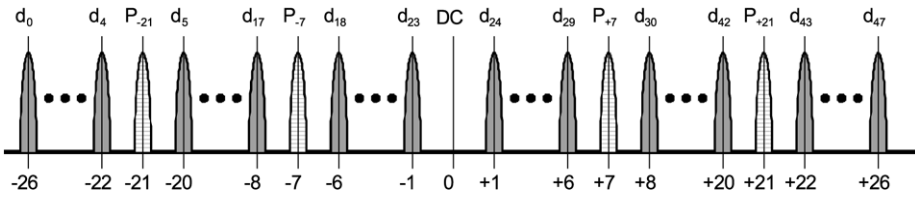


Figure 2.4: Subcarrier allocations for a 20 MHz channel [7]

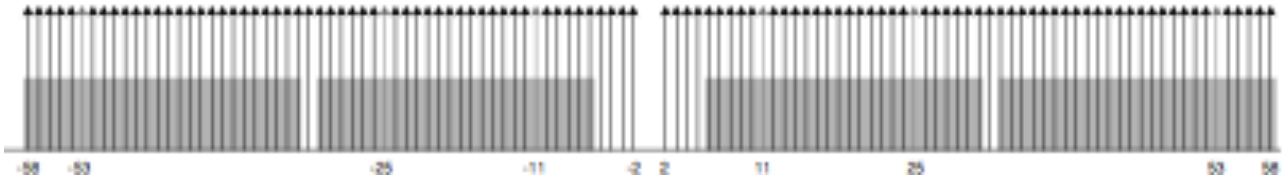


Figure 2.5: Subcarrier allocations for a 40 MHz channel [4]

e. MIMO

A system throughput rises linearly with each antenna that is added at both ends. Therefore, a total of 8 antennas, 4 at the transmitter and 4 at the receiver, will quadruple the systems throughput, thus allowing a total of 4 simultaneous 150 Mbit/s streams with the system achieving a total throughput of 600 Mbit/s [2][4]. There are certain trade-offs to applying on system though, such as an increased power consumption and cost, even though 802.11n includes a MIMO power-save mode that controls power consumption by using more than one data stream only on the additional performance becomes relevant.

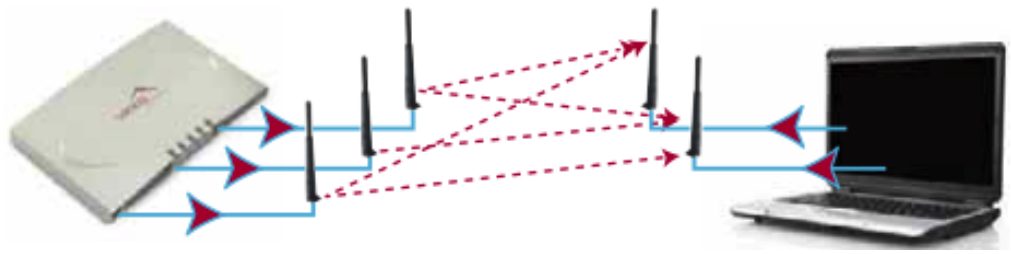


Figure 2.6: The MIMO Principle [8]

It is important to notice that 600 Mbit/s in raw throughput is not very relevant when it comes to data transmission. For instance, although the 802.11g can achieve a raw throughput of 54 Mbit/s, its throughput on the top layer can't go over 26 Mbit/s since the MAC overhead is over 50%. With the MAC overhead reduction applied on 802.11n due to aggregation and the standard achieving a PHY data rate of 600 Mbit/s, a MAC layer throughput above 400 Mbit/s can be attained.

Bibliography

[1] Matthew S. Gast, “802.11 Wireless Networks: The Definitive Guide”, Second Edition, 2005.

[2] E. Perahia and R. Stacey, “Next Generation Wireless LANs: Throughput, Robustness, and Reliability in 802.11n”, Cambridge University Press, 2008.

[3] “802.11n: Next Generation Wireless LAN Technology”, White Paper, Broadcom, April 2006.

[4] R. Van Nee, et al, “The 802.11n MIMO-OFDM Standard for WLAN and Beyond”, Wireless Personal Communications, Airgo Networks, 2006.

[5] Eldad Perahia, “IEEE 802.11n Development: History, Process, and Technology”, IEEE Communications Magazine, July 2008.

[6] Andrea Goldsmith, “Wireless Communications”, Stanford University, Cambridge University Press, 2005.

[7] http://www.emeraldinsight.com/journals.htm?articleid=1805845&show=html.

[8] MeRu Networks, “Wireless Without Compromise: Delivering the promise of IEEE 802.11n”.

Chapter 3

Orthogonal Frequency-Division Multiplexing

3.A) Introduction

This chapter presents an overview of digital communication systems with great focus on the modulation method used, Orthogonal Frequency-Division Multiplexing. Some of the advantages, disadvantages, applications and the history of OFDM are presented. The chapter is ended with a greater focus on the OFDM parameters and techniques used on the 802.11-2007 release.

3.B) Understanding Digital Communication Systems

A digital communication system involving OFDM includes transmission information in a digital structure from one point to another (source to sink) as shown in Figure 3.1.

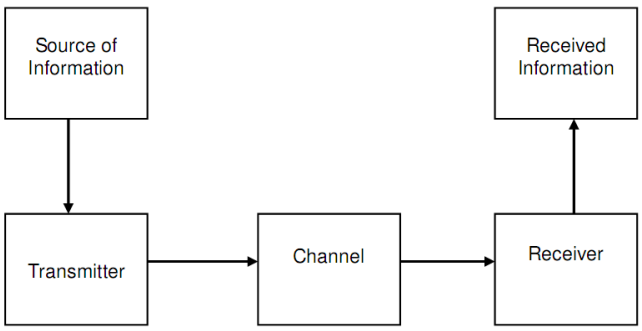


Figure 3.1: Digital Communication System

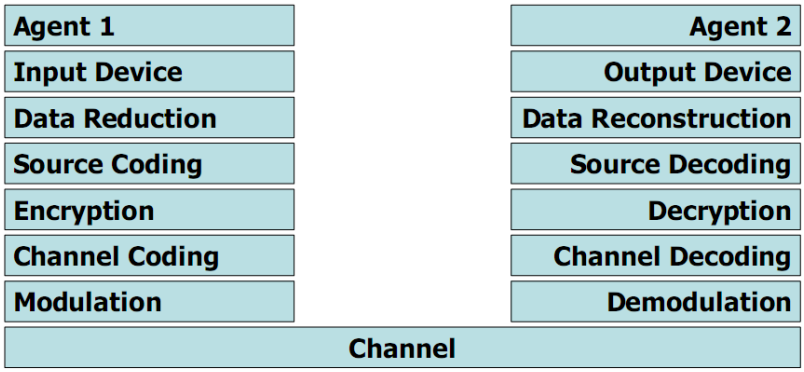


Figure 3.2: Simplified Communication System [24]

Transmitter, Receiver and Channel are the foundation of any communication system. Let's assume the simplified scheme of a communication system like the one on Figure 3.2, where two *agents* are communicating using a certain channel. The *agents* can either be persons, like in a standard phone call, but can also be machines like computers or mobile phones. The *channel* can either be a wired or wireless link. Before the generated data travels through the *channel* it has to be processed. At the transmitter side, the following (among others) functionalities can be distinguished. An *input device* is used to feed the data to the system like a keyboard or a video camera. In data reduction, all non-relevant data to the receiver is removed, such as the miter of an image. *Source Coding* techniques are applied so data is converted and/or compressed into a binary format. Compression can be lossy (e.g., audio, video, etc) or lossless (e.g., text). An *encryption* algorithm is applied (usually done by a scrambler) to protect data against eavesdropping. In *Channel Coding* the data is processed in order to raise its resistance against any nature of disturbance that can be found on the channel like noise, interference, distortion, etc. During transmission, these impairments attenuate the signal amplitude and distort the signal's phase. Also it is assumed that the noise present in the channel is white and has a Gaussian distribution. Most techniques consist in introducing some controlled redundancy in the message. Generally, the data shape presented at the *Channel Coding* output is not fit to be transmitted through the channel; therefore the data undergoes the *modulation* block where it's mapped into an appropriate waveform by a specified *modulation* method (like OFDM or QAM). *Modulation* techniques also contribute to raise the data resistance to impairments found in the channel, so it's important to consider both *modulation* techniques and *channel coding* techniques simultaneously and not separately. At the receiver side, the inverse operations are performed (adapted from [24]).

3.C) Introduction to Orthogonal Frequency-Division Multiplexing

Since 1999, 802.11 along with OFDM have been the tip of the movement in mobile data networking. OFDM can be seen as either a multi-carrier modulation technique or as a multiplexing technique, which just like *Frequency Division Multiple Access* (FDMA) divides the available spectrum into many subcarriers. In the wireless environment, these techniques are referred to as OFDM. In the wired environment, like *Asymmetric Digital Subscriber Lines* (ADSL), it is called *Discrete Multi Tone* (DMT). Orthogonality isn't always preserved in DMT though [7]. Compared to FDMA, it uses the spectrum even more efficiently by spacing the channels much closer together along with orthogonality in order to prevent interference between the closely spaced carriers. The main advantage of OFDM over single-carrier schemes is its ability to move through severe channel conditions without the use of complex equalization filters. Current OFDM systems are able to effectively avoid any *Inter Channel Interference* (ICI) due to the subcarriers orthogonality and erase any possibility of *Inter Symbol Interference* (ISI) occurring since the *Cyclic Prefix* (CP) was added in order to act a guard interval, but only as long the maximum channel delay in the channel doesn't overcome the guard interval [8].

3.D) History of OFDM

In the mid 60's, Chang [1] proposed a method where he overlaps orthogonal spectra to increase the efficiency of multicarrier systems by proving that the multipath problem can be solved without the need to reduce the data rate. The objective was to transmit simultaneous signals through a bandwidth-limited channel without any ICI or ISI taking place. Although this is considered to be by many the first leap taken on multicarrier systems, there are some known previous works like [3] and [4]. After conducting an analysis on Chang's work, Saltzberg [5] concluded that multicarrier systems must focus on reducing interference between channels instead of perfecting each individual signal.

In 1971, Weinstein and Ebert [2] proposed the Discrete Fourier Transform (DFT) method to perform both base band modulation and demodulation. DFT removes the banks of subcarrier oscillators and they used *Guard Interval*s to fight against ICI and ISI. However, the projected system did not attain perfect orthogonality between subcarriers over a dispersive channel.

It was in 1980 that Peled and Ruiz [6] introduced the *Cyclic Prefix*, solving the orthogonality problem. The guard interval was filled with a cyclic extension of the OFDM symbol.

From 1999 to 2004, the IEEE 802.1x groups adapt OFDM on the 802.11a, 802.11g and 802.16 (WiMAX) standards.

In October 2009, after 11 Draft Proposals by the 802.11n Task Group (composed by competitors TGn Sync, WWISE and MITMOT), the IEEE 802.11 Working Group publishes the 802.11n standard using MIMO-OFDM as its modulation technique.

3.E) OFDM Advantages

OFDM has quite a few advantages when compared to other types of modulation techniques that can be implemented in a wireless system. Some of the advantages are described below:

1. Bandwidth Efficiency

Bandwidth efficiency has been becoming along the years, the main key feature in high-speed communications given that all current and future devices already share a rather crowded range of frequencies. In OFDM, the frequency band containing the message is divided into parallel bit streams of lower-frequency subcarriers. IFFT/FFT operations ensure that these subcarriers are orthogonal to each other so that they can be separated at the receiver without any interference from adjacent carriers. Due to it, channels can be spaced much closer together, allowing for more efficient use of the spectrum compared to common *Frequency Division Multiplexing* (FDM) (Figure 3.3).

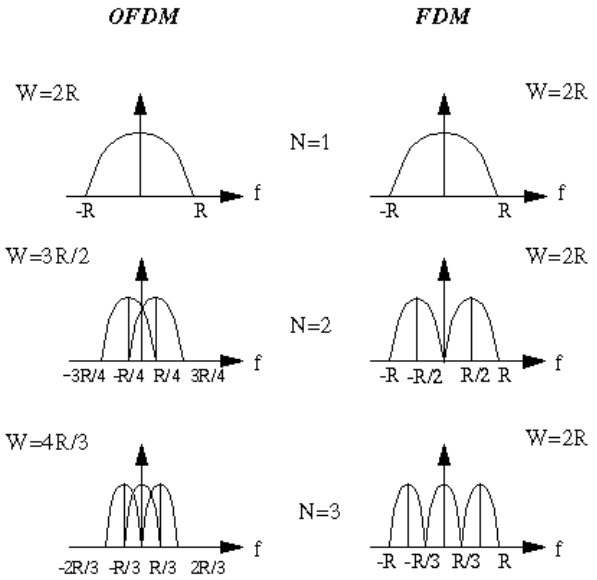


Figure 3.3: OFDM (left) versus FDM (right) [22]

This advantage of OFDM does not occur in FDMA; where up to 50% of the total bandwidth is wasted due to the extra spacing between channels as shown on Figure 3.4.

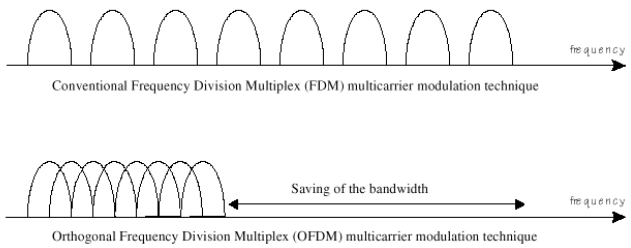


Figure 3.4: Spectrum use of a classic FDM signal (top) when compared to an OFDM signal (bottom) [23]

2. Combating Inter Symbol Interference (ISI)

ISI becomes a limitation in data rate capability since the delay time caused by multipath remains constant. Higher data rates require smaller symbol periods and when these symbol periods become smaller than the delay spread, ISI occurs. In order for the CP to be effective its length must be at least equal to the length of the multipath channel. Delay spread is considered as the interval for which a symbol remains inside the multipath channel. Since low symbol rate modulation schemes suffer less from ISI, it is advantageous to transmit multiple low rate streams instead of a single high rate stream.

Since the duration of each symbol is long, it is feasible to insert a CP between OFDM symbols in order to eliminate ISI. The technique consists in replicating part of the time-domain wave shape from the end of the symbol to the front, creating a guard period between symbols. The CP also eliminates the need for a pulse-shaping filter and reduces the sensitivity to time synchronization problems [9]. Although it is usually discarded at the receiver it serves its purpose, also simplifying the channel equalization in the demodulator besides fighting ISI.

3. Frequency Selective Fading and Burst Errors

Since the channel is divided into narrowband flat fading subchannels, burst errors caused by a deep fade or impulse interference are randomized. So instead of having several adjacent symbols being totally destroyed, many symbols are slightly destroyed like shown on Figure 3.5. So symbols lost due to the frequency selectivity of the channel can then be recovered through FEC codes and Interleaving techniques.

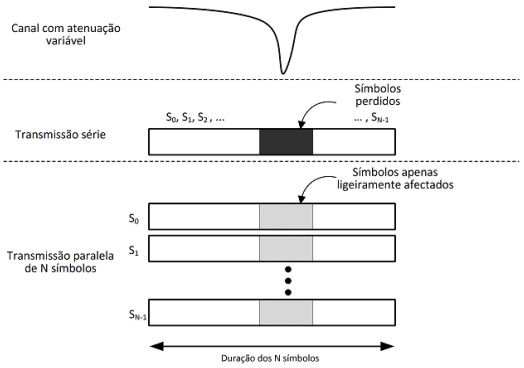


Figure 3.5: Attenuation outcome on serial and parallel transmissions [10]

4. Fighting Inter Channel Interference (ICI)

Two factors cause ICI, namely frequency offset, like the Doppler Shift, and time variation. ICI becomes even more of an issue due to the small separation between subcarriers. OFDM fights multipath with subcarrier orthogonality. If the delay spread is smaller than the CP there will be no ICI [19].

5. ***Orthogonal Frequency-Division Multiple Access*** **(OFDMA)**

OFDMA consists on a technique that takes profit of OFDM on a multi-user environment [16][17]. Each user is assigned a defined set of subcarriers allowing simultaneous data transmission over several users like shown on Figure 3.6.

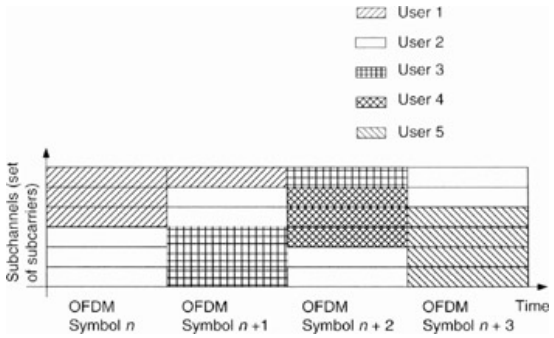


Figure 3.6: User assignment of a set of subcarriers on OFDMA [21]

3.F) OFDM Disadvantages

Although it excels in its fight against attenuation, interference and other channel impairments, OFDM isn't free of weaknesses. Some of those weaknesses are presented below:

1. ***Peak-to-Average Power Ratio*** **(PAPR)**

Unlike a *Frequency Modulated Continuous Wave* (FMCW) signal, OFDM's signal information presents itself in the signal's amplitude variations like shown on Figure 3.7 so it is very important that those variations are kept intact. If the signal's amplitude suffers modifications in any way, the transmitted signal's FFT will no longer represent the original frequency characteristics.

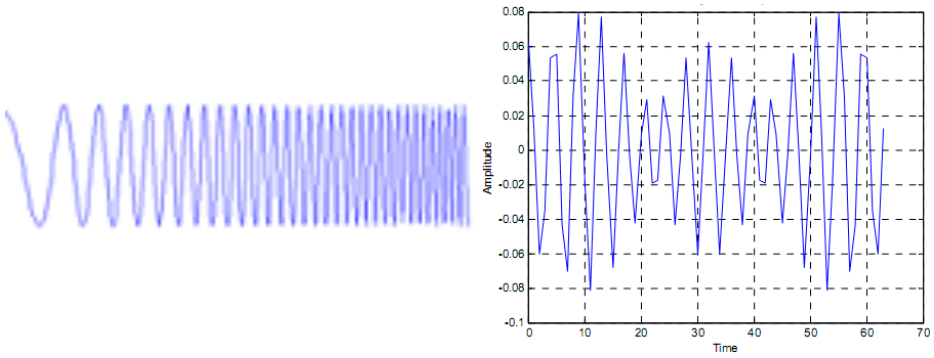


Figure 3.7: FMCW signal (left) and OFDM signal (right)

An OFDM signal is the sum of many separate sinusoids. In a worst case scenario, assuming all the sinusoids add up constructively, extremely large amplitude peaks may occur bringing the *peak-to-average power ratio* (PAPR) to a higher level. This particularity becomes even more problematic when the amplitude peaks go outside the dedicated band. This results in an increasing complexity towards the conversion systems (ADC's and DAC's) that will need higher resolutions, making them more expensive and requiring a higher power usage in order to deal with the high amplitude peaks. Common PAPR reduction techniques consist in distortion techniques, like clipping and peak windowing [11]; codification techniques, based in OFDM symbol coding so that these show a lower PAPR [12]; scrambling techniques, which shift the content of a OFDM signal in order to obtain a symbol with an acceptable PAPR [13] and predistortion techniques which consist in predistorting the signal to compensate for the amplifier nonlinearity [14].

2. Synchronization

The need to maintain the subcarriers orthogonality during transmission makes OFDM signals particular sensitive to frequency differences between the local oscillators on both transmitter and receiver. These differences are also increasingly dependent on the number of subcarriers in use but decrease with gap enlargement between them. This offset causes attenuation, rotation and ICI between subcarriers. The relative movement between transmitter and receiver is bound to cause Doppler frequency shifts leading to a lack of synchronization between subcarriers. Consequently, time and frequency synchronization between transmitter and receiver is of the utmost importance. This is achieved by using known pilot tones inserted into the OFDM signal and/or attaching fine frequency timing tracking algorithms within the OFDM signal's cyclic extension.

3.G) Applications of OFDM

OFDM has become the choice of several past, current and future communications systems and standards all over the world. The first standard to use OFDM as its modulation method was the *Digital Audio Broadcast* (DAB) system, introduced in the beginning of the 1990-decade [15]. Several OFDM based systems followed DAB; like *Digital Video Broadcast – Terrestrial* (DVB-T), which uses a FFT size of 2048 or 8192 points and achieves a typical bit rate of 24 Mbit/s. In wired applications, OFDM can be found in ADSL and *High bit-rate Digital Subscriber Lines* (HDSL) systems. Powerline also adopted OFDM in order to extend Ethernet connections and due to its resilience to dispersive channel and narrowband interference in such a noisy channel like electrical wiring. IEEE 802.11g/n, WiMAX and HyperLAN/2 are likely the most well known standards that adopted OFDM recently.

Standard	ETSI DAB	ETSI DVB-T	IEEE 802.11a	IEEE 802.11.n
Ratified Year	1995	1997	1999	2009
Number of subcarriers	192, 384, 768, 1536	1705, 6817	52	52
Mapping Schemes	π/4-DQPSK	QPSK, 16-QAM, 64-QAM	BPSK, QPSK, 16-QAM, 64-QAM	BPSK, QPSK, 16-QAM, 64-QAM
FFT Size	256, 512, 1024, 2048	2048, 8192	64	64
Channel Spacing (MHz)	1.712	6, 7, 8	20	20, 40
Bit Rate (Mbit/s)	0.576 – 1.152	4.98 – 31.67	6 – 54	6.5 – 600
Frequency Range (GHz)	0.17 – 0.24; 1.45 – 1.49	0.47 – 0.86; 0.17 – 0.23	5.15 – 5.825	2.417 – 2.7; 5.0 – 5.7

Table 3.1: Four OFDM based standards and some of their parameters

3.H) OFDM on 802.11-2007

Although major changes have taken place on the IEEE 802.11 standard due to the 802.11n amendment, the latest official release by the group still the remains the IEEE 802.11-2007 that includes amendments a, b, d, e, g, h, i and j released on July 2007. Therefore the OFDM signal assembly is based on this latest release and not the one that will be launched including the 802.11n amendment. Figure 3.8 presents a typical structure

of an OFDM transmitter and receiver system where some of the main blocks are incorporated.

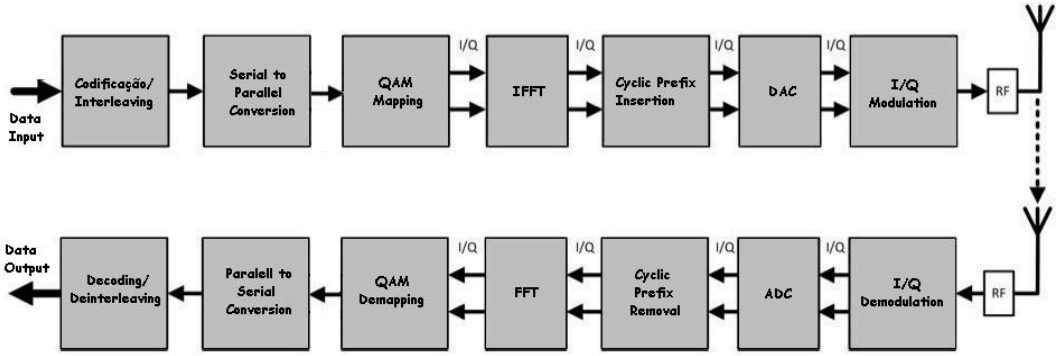


Figure 3.8: Simplified Typical OFDM System

When compared to a typical OFDM transmitter, the difference between this implementation and a typical one is the absence of an error correction coding and interleaving system along with the processing needed to make sure the signal has the appropriate parameters to be transmitted, such as filtering and PAPR conditioning. Designing an OFDM that meets every rule defined by the standard is task that surpasses the scope of this project. Also, the introduction of a channel encoder and/or interleaver didn't seem appropriate for this project given that in the Matlab simulation no channel impairments are introduced, such as noise, ISI, ICI, among others. The hardware co-simulation is performed through a wired channel, thus not introducing relevant noise, frequency shifts or other kind of channel impairments that could be tested with relevance. The remaining obsolete blocks would require a deep study in not only such technique, but also on the standard itself.

According to the standard's latest release [18], the OFDM signal assembled on the PHY layer should have 64 subcarriers in order for the *Inverse Fast Fourier Transform* (IFFT) to be performed. 48 subcarriers are fully dedicated to the data to be transmitted, 4 devoted to pilot tones, 1 DC subcarrier that is kept at zero and 11 subcarriers that are kept unused which will act as the guard intervals (Figure 2.4). From these last 11 subcarriers, 6 belong to the guard interval on the lower frequencies and 5 to the guard interval on the higher frequencies.

Among the 4 possible modulation methods provided by the standard, 16-QAM OFDM will be used. Therefore, 4 coded bits per subcarrier will exist with a total of 192 coded bits per OFDM symbol.

802.11 supports one of three possible channel spacings: 20 MHz, 10 MHz or 5 MHz. Assuming a typical channel spacing of 20 MHz, the subcarrier frequency spacing will be 312.5 kHz (=20 MHz/64) with a total IFFT period of 3.2 μs (=1/0.3125 MHz).

Once the IFFT is performed, the CP is added with a size of 25% of the size of the original symbol. So the total duration of an OFDM frame is 4.0 μs (= 3.2 + 0.8) or 80 chips, 64 for data and 16 for the cyclic prefix, corresponding to an efficiency of 80% like shown on Figure 3.9.

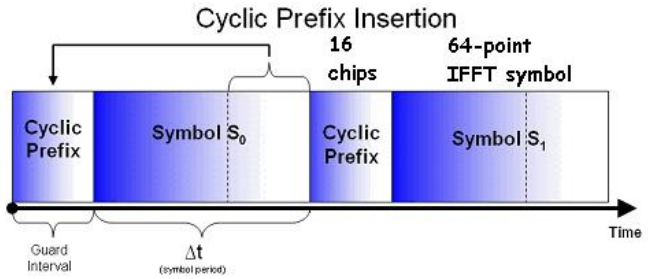


Figure 3.9: Frame Structure of an OFDM symbol [20]

Each 802.11 frame contains a variable number of OFDM signals. The number of data octets that can be transferred with a single frame ranges from 1 to 4095 [18], meaning each frame can possess a total of 32760 bits. Since 16-QAM is applied and each OFDM symbol has a total of 192 data bits per symbol, a maximum of 170 OFDM symbols can be inserted on each frame. For the project, each frame will have 5 OFDM symbols.

Table 3.2 shows some of the system parameters relevant for this project.

Parameter	Value
N_{SD}: Number of data subcarriers	48
N_{SP}: Number of pilot subcarriers	4
N_{FFT}: IFFT/FFT size	64
Channel spacing	20 MHz
Δ_F: Subcarrier frequency spacing	312.5 kHz (20 MHz/N_{FFT})
T_{FFT}: IFFT/FFT period	3.2 µs ($1/\Delta_F$) or 64 chips
T_{CP}: CP period	0.8 µs ($T_{FFT}/4$) or 16 chips
T_{SYM}: OFDM symbol period	4 µs ($T_{FFT}+T_{CP}$) or 80 chips
Number of guard bands subcarriers	6 (lower band) and 5 (higher band)
Guard bands indexes	-32 to -27 and 27 to 31
Pilot tones Indexes	-21, -7, 7, 21
Modulation type	16-QAM OFDM
N_{BPSC}: Coded bits per subcarrier	4
N_{CBPS}: Coded bits per OFDM symbol	192
N_{DBPS}: Data bits per OFDM symbol	192
OFDM symbols per frame	5
Coded bits per frame	960 ($5*N_{CBPS}$)
Data bits per frame	720 ($5*N_{DBPS}$)
T_{FRAME}: Frame duration	20 µs ($5*T_{SYM}$)

Table 3.2: System parameters relevant for this project [18]

Bibliography

[1] R. W. Chang, “Synthesis of band limited orthogonal signals for multichannel data transmission”, Bell Systems Technical Journal, pp. 1775-1796, December 1966.

[2] S. B. Weinstein and P.M. Ebert, “Data transmission by frequency-division multiplexing using the discrete Fourier transform”, IEEE Transactions on Communication Technology, vol. COM-19, pp. 628-634, October 1971.

[3] J. L. Holsinger, “Digital communication over fixed time-continuous channels with memory, with special application to telephone channels,” PhD thesis, Massachusetts Institute of Technology (MIT), 1964.

[4] Robert G. Gallager, “Information Theory and Reliable Communications”, Wiley, January 1968.

[5] B. R. Saltzberg, “Performance of an Efficient Parallel Data Transmission System”, IEEE Transmission Communications, pp. 805-811, December 1967.

[6] A. Peled and A. Ruiz, “Frequency Domain Data Transmission using Reduced Computational Complexity Algorithms”, IEEE International Conference on Acoustics, Speech, and Signal Processing, pp. 964-967, Denver, April 1980.

[7] Dusan Matiae, “OFDM as a possible modulation technique for multimedia applications in the range of mm waves,” TUD-TVS, October 1998.

[8] Internet, “Orthogonal Frequency Division Multiplex”, Tutorials in Communications Engineering, http://www.complextoreal.com/tutorial.htm, 2010.

[9] Magis Networks, “Orthogonal Frequency Division Multiplexing (OFDM) Explained”, White paper, Inc. 2001.

[10] Afonso Resende Silva, “Geração de Sinais Vectoriais Baseada em FPGA”, Masters Thesis, Universidade de Aveiro, 2008.

[11] K. R. Panta and J. Armstrong, “Effects of Clipping on the Error Performance of OFDM in Frequency Selective Fading Channels”, IEEE Transactions on Wireless Communications, Vol. 3 Number 2, March 2004.

[12] Filbert H. Juwono and D. Gunawan, “PAPR reduction using Huffman coding combined with clipping and filtering for OFDM transmitter”, CITISIA 2009, pp 344-347, July 2009.

[13] Lei Wang, Kyongkuk Cho, Dongweon Yoon, and S. Kyu Park, “PAPR Reduction of OFDM Signals Using Deliberate Clipping and Pre-scrambling Technique”, Hanyang University, Seoul, Korea.

[14] Yan Tang, Keang-Po Ho and William Shieh, “Coherent Optical OFDM Transmitter Design Employing Predistortion”, IEEE Photonics Technology Letters, Vol.20 Issue 11, pp 954-956, June 2008.

[15] “Radio Broadcasting Systems; Digital Audio Broadcasting (DAB) to mobile, portable and fixed receivers”, European Standard (Telecommunication Series), Final Draft ETSI EN 300-401, January 2006.

[16] A. Paulraj, R. Nabar, D. Gore, “Introduction to space-time wireless communications”, First Edition, Cambridge University Press, 2003.

[17] H. Liu, G. Li, “OFDM-based broadband wireless networks: design and optimization”, First Edition, John Wiley & Sons, 2005.

[18] IEEE Computer Society, “IEEE Standard for Information technology—Telecommunications and information exchange between systems—Local and metropolitan area networks— Specific requirements – Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications”, IEEE Std 802.11™ – 2007, 12 June 2007.

[19] Andrea Goldsmith, “Wireless Communications”, Stanford University, Cambridge University Press, 2005.

[20] National Instruments, “OFDM and Multi-Channel Communication Systems”, http://zone.ni.com/devzone/cda/tut/p/id/3740.

[21] WiMAX. Technology for Broadband Wireless Access, http://flylib.com/books/en/4.136.1.35/1.

[22] Pierre Coulon, “Principles of Modulation in Wireless Communications”, October 1999.

[23] http://www.wirelesscommunication.nl/reference/chaptr05/ofdm/ofdmhist.htm.

[24] Jos H. Weber, “Error-Correcting Codes”, January, 2007.

Chapter 4

Field-Programmable Gate Array

4.A) Introduction

Due to the importance of the FPGA in the project, this chapter is fully dedicated to it. An overview of its architecture and characteristics is performed, followed by a comparison to ASICs and CPLDs and describing why the FPGA is the best choice for this project among other kind of platforms. The chapter ends with a basic overview of *Xilinx's System Generator for DSP* and some of the devices built by Xilinx for *Digital Signal Processing* (DSP).

4.B) Architecture and Characteristics

An FPGA is a device consisting of three key components: *Configurable Logic Blocks* (CLB), a distributed interconnection programmable structure and programmable I/O blocks (Figure 4.1). It is built in such a way that a user with the aid of a CAD environment can configure it. Its main advantage lies in its reprogrammability, which allows itself to be reprogrammed at any moment in order to implement a system according to the end-user needs, unlike integrated circuits designed for a specific purpose. An FPGA is considered to have a fine-grained architecture because it contains numerous tiny indivisible logic blocks than can reach up to 100,000, making it more suitable for large. Although it can implement just about any hardware design, such design is bound to end up in an ASIC due to reasons that will be seen later in this chapter.

There are two varieties of FPGAs on the market nowadays: SRAM-based FPGAs and anti-fuse based FPGAs. Xilinx and Altera are the market leaders in number of users when it comes to SRAM-based FPGAs and these are the ones on focus here. Xilinx was the first company to create a commercial FPGA, the XC2064 model, back in 1985 [1].

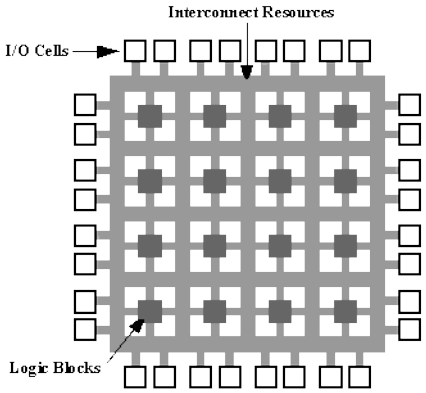


Figure 4.1: Structure of an FPGA [19]

Although different FPGA models incorporate different types of CLBs with different components and purposes, CLBs are usually composed of *Lookup Tables* (LUTs) with 4 or 6 inputs [2], carry circuits for arithmetic operations purposes, registers and multiplexers. A LUT is essentially a memory made from SRAM cells with a width of a single bit, where its address lines provide the inputs and the output from the memory is the LUT output. Since it only works with zeros and ones, R inputs correspond to a 2^R memory and can perform any logic function once the logic function's truth table is inserted into the memory. Due to all the LUTs, the FPGA architecture becomes has more flexible than of a CPLD, making FPGAs better in register-heavy and pipelined applications, as well as larger logic designs.

There are two possible implementations currently used in FPGAs for LUTs. With pipelining, higher internal frequencies can be achieved with the downside of a higher latency. The other option is by using two levels of logic in which a lower latency can be achieved, but higher frequencies are harder to reach.

Designs implemented in a FPGA demand different types of connections between the several CLBs required for its construction. Its distributed interconnect structure is another key feature of the FPGA, as well as what turns the FPGA into a reconfigurable device. The structure consists of horizontal and vertical long lines spanning throughout the device in order to apply the connections between the CLBs and the I/O ports. CAD environment tools usually keep the interconnecting task hidden to users unless specified otherwise, thus significantly reducing design complexity [2].

I/O blocks encase the inner structure, each one providing an individually selectable input, output or giving bi-directional access to a general-purpose I/O pin on the exterior of the FPGA encasement. They can also hold tri-state capability.

Among FPGA applications there is Error Control Coding, *Code Division Multiple Access* (CDMA) communications, Token Ring Networks, *Additive White Gaussian Noise* (AWGN) channels, including others.

4.C) ASIC, CPLD or FPGA?

In Chapter 1, a brief description of the PLDs considered for the project was carried out. In this subchapter, a deeper overview of each device will be performed and compared to the FPGA.

Whether it's a FPGA, an ASIC or a CPLD, each device will have advantages and disadvantages depending on several factors such as the kind of system to be implemented, available time to implement it, unit price, reconfigurability and what available tools are there to implement it, among others.

i) FPGAs versus CPLDs

Although CPLDs and FPGAs are normally produced by the same enterprises, both devices have several differences between them.

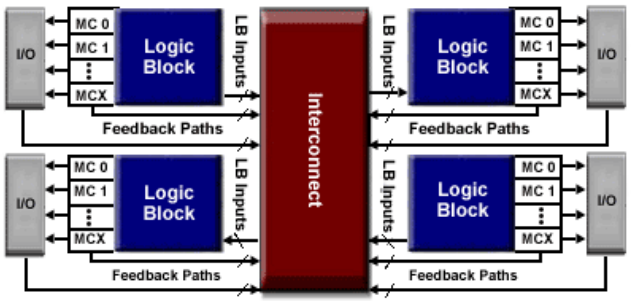


Figure 4.2: Internal Structure of a CPLD [10]

CPLDs' are made-up from a small number (typically reaches a few hundreds) of very big logic blocks. Although this characteristic gives them a faster response time, it has a lot less flexibility when compared to a typical FPGA's architecture composed of tiny logic blocks with a number of flip-flops that can reach a million. This makes CPLDs not only more appropriate for relatively small designs, while FPGAs can handle larger ones; but their structure also makes them faster and more predictable. The difference in both architectures makes the FPGA better-purposed for register-heavy and pipelined applications, whereas CPLDs will be better where the processing of input data streams must be performed at a very fast pace for instance. FPGAs also have a higher ratio of gates in a given area and cost habitually less than CPLDs, thus making FPGAs the go-to choice for larger designs [3]. A CPLD's architecture also allows it to reach high frequencies, which isn't possible when using two levels of logic in a FPGA, however its operation speed as a tendency to decrease with the increase in PLD density for CPLDs.

The connections inside the logic blocks (PLDs) shown in Figure 4.2 and the interconnect matrix used by the CPLD are different. While the logic blocks interconnect structure is fully connected, the CPLD structure may not be; thus making some hardware designs simply not possible within a given CPLD, even though there are enough gates available. On the other hand, FPGAs make use of their distributed interconnect structure by having specialized routing resources to implement arithmetic functions more efficiently.

One of the main differences between the FPGA used on the project and a CPLD is the existence of a non-volatile memory on-chip in the CPLD. FPGAs use volatile SRAM technology in order to load programs into the FPGA, therefore these programs are lost if the FPGA is powered out, unlike EEPROM-based devices or flash-based devices such as CPLDs. However this distinction is starting to become less relevant given that the latest FPGA models start to offer an embedded memory.

CPLDs are usually used *Local Area Network* (LAN) controllers, cache memory controllers, graphics controllers, *Universal Asynchronous Receiver/Transmitters* (UARTs) and generally in applications that require a large quantity of AND/OR gates but not a large number of flip-flops [11].

ii) FPGAs versus ASICs

According to Moore's Law [12] the speed of computer systems as a tendency to double roughly every 18 months. So what's the point in working with hardware-based designs such as ASICs if we can wait a while for a general-purpose processor to reach the market? If something can be done with an FPGA today there is little point in waiting. The power brought by CAD environments has been growing over the years, for that reason

designing is becoming easier and faster all the time, therefore we can have the application we want sooner. Also, even if general-purpose CPUs become fast enough to implement a particular design, there may still be environments where an FPGA-based implementation is still more appropriate. It is also important to realize that Moore's Law can be applied to both FPGA and CPU; so when CPUs become fast enough to implement a certain algorithm, FPGAs should be capable of implementing even more complexly intensive designs [13]. FPGAs may even surpass general-purpose CPUs in performance for some types of computations [14].

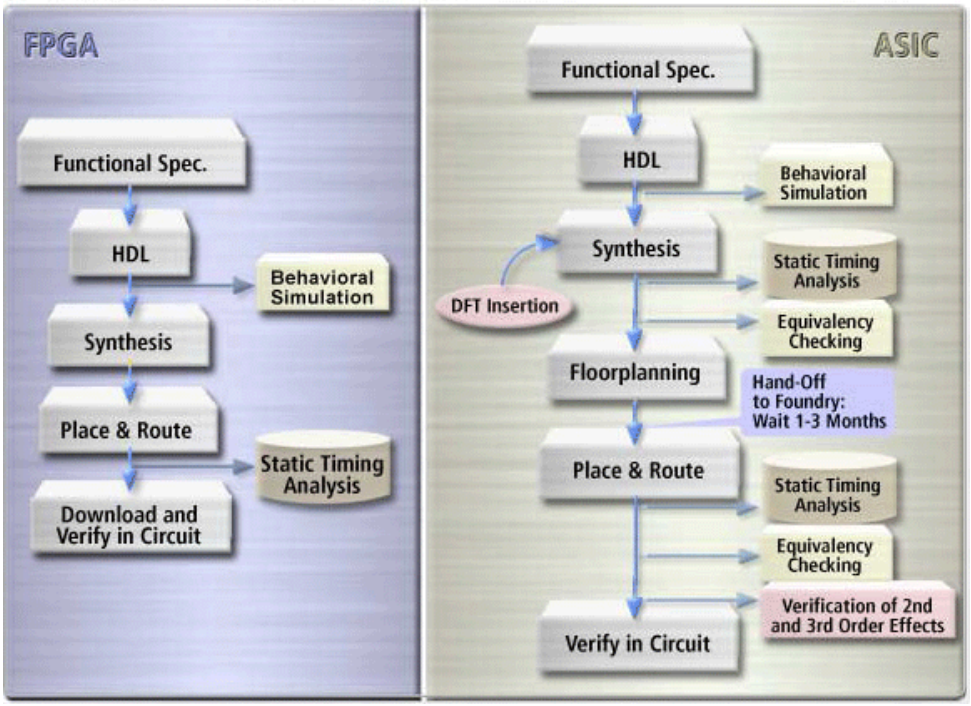


Figure 4.3: FPGA and ASIC Design Flows [4]

Regarding time consumption, Figure 4.3 shows that an FPGA implementation avoids many steps usually necessary for an ASIC implementation and due to that, it trims down the time needed for the algorithm to be ready. The main difference on a FPGA implementation is the suppression of the necessity to wait for the layout blueprint of the circuit as well as the whole manufacturing process. Skipping these two steps not only saves time and heaps of manpower, but also *Non Recurring Expenses* (NRE) which are typically associated when designing an ASIC, although when it comes to a single application and a high number of devices, ICs mass-production ends up being way cheaper and implementations tend to have a lower power consumption per unit when compared to FPGAs. So ultimately, pricewise, the developing process cost will frequently

be lower with FPGAs, but if it's the purpose of the design to reach mass-production, choosing an ASIC will be the best choice. Mass-production also has its risks though; an example can be for instance when a mistake is made during an ASIC design process but it's only detected after mass-production of the circuit.

Hardware upgrading or a different purpose application is another aspect that an ASIC can't keep up with. Full reconfigurability is key on a FPGA, not only allowing the implementation of totally different designs on the same FPGA, but also fixing errors made during designs.

iii) Platform Choice

Taking into account all of the above, using an FPGA on a project like the one presented on this thesis allows the implementation of the system in a faster and fully reconfigurable way in case design mistakes occur during the designing process. Although power consumption is important, it isn't the purpose of this work to mass-produce the algorithm shown ahead. It also allows the FPGA in question to be used in the future for either a totally different algorithm or for improving the developed algorithm.

4.D) Xilinx System Generator for DSP

The purpose of this sub-chapter is to give a brief description of the main CAD environment tool, developed by Xilinx that was used for the development of the project.

Xilinx's System Generator for DSP (SysGen) is along ISE Design Suite the main tool for design development and the most used CAD environment on the project. SysGen consists of a library package that is inserted into *Simulink*; a tool for modeling, simulating and analyzing multi-domain dynamic systems inserted in *Matlab*; that is embedded into Matlab. Simulink's interface is a graphic block diagramming tool along with a given number of included libraries where SysGen is another one of those libraries with 3 subsections called *blocksets* [15] containing over 90 diverse multi-purpose blocks. Some of these blocks are shown in Figure 4.4. Just like ISE Design Suite, steps like synthesis and place & route are performed automatically and implanted into a programming file that will run on the FPGA [16].

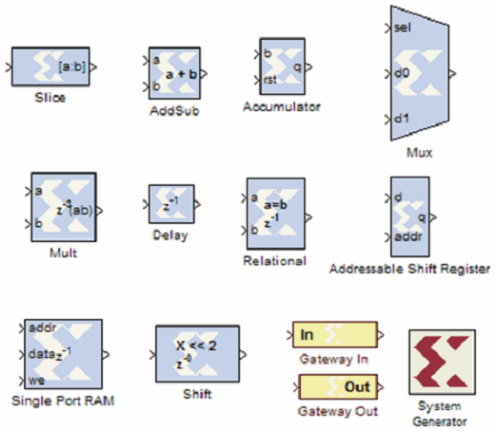


Figure 4.4: System Generator Blocks [5]

The 3 *blocksets* introduced into *Simulink* by *SysGen* are the following: *Xilinx Blockset, Xilinx Reference Blockset* and *Xilinx XtremeDSP Kit Blockset*.

Among all the blocks used on the design, the majority of them can be found on the first blockset, *Xilinx Blockset.* This particular blockset is divided into 9 subsections. These subsections are listed below along with a brief description of the block types and some blocks that can be found in each subsection [15]:

- *Basic Elements*, standard building blocks for digital logic; e.g.: Parallel to Serial, Mux, Register, Reinterpret, Convert, etc.
- *Communication*, forward error correction and modulator blocks, commonly used in digital communications systems; e.g.: Depuncture, Reed-Solomon Decoder 7.1, Viterbi Decoder 7.1, etc.
- *Control Logic*, blocks for control circuitry and state machines; e.g.: Constant, Counter, FIFO, Slice, Dual Port Ram, etc.
- *Data Type*, blocks that convert data types and gateways; e.g.: Concat, Scale, Shift, Gateway Out, Gateway In, etc.
- *DSP*, Digital Signal Processing (DSP) blocks; e.g.: CORDIC 4.0, CIC Compiler 2.0, DSP48, *Linear Feedback Shift Register* (LFSR), etc.
- *Index*, Contains all the blocks present on the blockset.
- *Math*, blocks that implement mathematical functions; e.g.: AddSub, Accumulator, Inverter, Negate, etc.

- *Memory*, blocks that implement and access memories; e.g.: ROM, Delay, Addressable Shift Register, Single Port RAM, etc.
- *Shared Memory*, blocks that implement and access Xilinx shared memories; e.g.: Shared Memory Read, From Register, From FIFO, To FIFO, etc.
- *Tool*, "Utility" blocks; e.g.: ChipScope, Clock Probe, FDATool, ModelSim, System Generator, etc.

It should also be noted that four Simulink blocks (Mux, Goto, Demux, From) exist that can be mapped into hardware if needed be, given that they are fully supported by SysGen. Other Simulink blocks can be included in a Xilinx design for simulation purposes, but such blocks will not be mapped into hardware.

The second blockset, *Xilinx Reference Blockset*, consists of more complex blocks that can be built from blocks from the first blockset. They put up a facade to a higher complexity system, making it thus easier to use. This blockset is divided into 5 subsections, which are listed below along with a brief description of the block types and some blocks that can be found in each subsection [15]:

- *Communication,* blocks dedicated to AWGN channels simulation; e.g.: BSPSK AWGN Channel, Convolutional Encoder, etc.
- *Control Logic*, blocks for implementing state machines; e.g.: Mealy State Machine, Moore State Machine, etc.
- *DSP*, blocks for implementing filters; e.g.: CIC Filter, Interpolation Filter, n-tap MAC FIR Filter, etc.
- *Imaging*, blocks dedicated to image processing; e.g.: 5x5 Filter, Virtex Line Buffer, Virtex2 5 Line Buffer and Virtex2 Line Buffer.
- *Math*, blocks to implement CORDIC algorithms for mathematical functions; e.g.: CORDIC DIVIDER, CORDIC LOG, CORDIC SQRT, etc.

The third and last Xilinx's blockset, *Xilinx XtremeDSP Kit Blockset*, inserts six blocks into Simulink; all of them related to Nallatech's *XtremeDSP Kit-IV* in this case, but all six blocks are shared among most boards. All blocks on this blockset correspond to physical components of the boards such as DACs and ADCs. It was an extremely important blockset on Chapter 6 since it allowed the emulation of the real behavior of a DAC when

the hardware co-simulation was performed. The six blocks that can be found on this blockset are the following:

- *DAC1 and DAC2*, two Digital-to-Analog converters representing the two DACs on the board;
- *ADC1 and ADC2*, two Analog-to-Digital converters representing the two ADCs on the board;
- *External RAM,* allows components to connect to the external ZBT SRAM present on Nallatech's BenAdda board;
- *LED Flasher,* block that allows interaction with the tri-color LEDs present on Nallatech's BenAdda board when performing a co-simulation.

4.E) Xilinx Devices for DSP Purposes

Family	Virtex-4		Virtex-5 SXT		Virtex-6 SXT		Spartan-6 LXT	
Model	4VSX25	**4VSX35**	5VSX50T	5VSX95T	6VSX315T	6VSX475T	6SLX45T	6SLX100T
Logic Cells	23.040	**34.560**	52.224	94.208	314.880	476.160	43.661	101.261
Max DSP Frequency (MHz)	500	**500**	550	550	550	550	390	390
XtremeDSP Slices	128	**192**	288	640	1344	2016	58	180
Block RAM (Kbits)	160	**240**	780	1.520	5.090	7.640	401	976

Table 4.1: Current Xilinx DSP dedicated models (in bold, the one used for this project) (adapted from [6])

Although the Virtex-4 SX35 was only conceived in 2004 [7], Xilinx has been upgrading its models and creating ones for other purposes such as aerospace and medical applications along the years [8].

Xilinx's Virtex-5 platform creation in 2006 [9] gave a considerable boost to logic performance with its 65-nm ExpressFabric™ technology when compared to Virtex-4's 90 nm devices. It delivers more than 500 GigaMACs (10^9 Multiply-Accumulate operations per second) and brings up to 1000 XtremeDSP slices, while consuming 35 percent less dynamic power when compared to previous 90-nm devices.

Among Xilinx's latest offers are the Virtex-6 SXT devices, providing over 1 TeraMACs with up to 2016 XtremeDSP slices, while reducing power consumption by 65

percent and achieving a bandwidth above 1 Tbps. The Spartan-6 platform is the latest upgrade for the Spartan family; a family that has been dedicated since its creation to low-cost DSP and memory; providing users with a good price/performance relation.

Looking at Table 4.1, it can be easily seen the increase in logic cells along the years. Due to the increase in the amounts of CLBs used in an FPGA, along with size reduction and chip price, the price of an individual NAND or NOR gate is rapidly approaching zero.

4.F) Altera Solutions for DSP

Altera is the other major FPGA manufacturer on the market nowadays offering DSP solutions similar to Xilinx such as FPGAs, CAD environments, CPLDs and ASICs.

The company brings 3 series of FPGAs to the market. The Stratix family offers high-end FPGAs and can be compared to Xilinx's Virtex series delivering state-of-the-art performance. The Arria series offers low cost on transceiver-based applications similarly to Xilinx's Spartan series. Technology processes range from 28 to 60 nm. The latest Stratix model brings innovative blocks incorporating 18x27, 36x36 and floating point multipliers [17].

The software provided by Altera is entitled *DSP Builder* and is incorporated into Matlab's Simulink like Xilinx's System Generator for DSP.

Bibliography

[1] Xilinx Inc., “Our History”, http://www.xilinx.com/company/history.htm, 2010.

[2] Xilinx Inc., “FPGA Basics”, http://www.xilinx.com/company/gettingstarted/index.htm#fpgabasics, 2010.

[3] Barr, Michael. "Programmable Logic: What's it to Ya?”, Embedded Systems Programming, June 1999, pp. 75-84.

[4] Xilinx Inc., “FPGA vs. ASIC”, http://www.xilinx.com/company/gettingstarted/index.htm#fpgavsasic, 2010.

[5] Xilinx Inc., “System Generator for DSP Getting Started Guide”, Release 10.1, March 2008.

[6] Xilinx Inc., “DSP Solutions Using FPGAs”, http://www.xilinx.com/products/design_resources/dsp_central/grouping/fpgas4dsp.htm, 2010.

[7] Xilinx Inc., “Virtex-4 Family Overview”, http://www.xilinx.com/support/documentation/data_sheets/ds112.pdf, 2010.

[8] Xilinx Inc., “Market Solutions”, http://www.xilinx.com/esp/index.htm, 2010.

[9] Xilinx Inc., “Virtex-5 Family Overview”, http://www.xilinx.com/support/documentation/data_sheets/ds100.pdf, 2010.

[10] Xilinx Inc., “CPLD Basics”, http://www.xilinx.com/company/gettingstarted/index.htm#cpldbasics, 2010.

[11] Steven T. Karris, “Digital Circuit Analysis and Design with Simulink Modeling and Introduction to CPLDs and FPGAs”, Second Edition, 2007.

[12] Ilkka Tuomi, “The lives and deaths of Moore’s law”, http://firstmonday.org/htbin/cgiwrap/bin/ojs/index.php/fm/issue/view/151, 2002.

[13] W. James MacLean, “An Evaluation of the Suitability of FPGAs for Embedded Vision Systems”, 2005.

[14] Keith D. Underwood, “FPGAs vs. CPUs: Trends in peak floating point performance”, 12th ACM International Symposium on Field Programmable Gate Arrays, pages 171–180, February 2004.

[15] Xilinx Inc., “System Generator for DSP Reference Guide”, Release 12.2, July 2010.

[16] Xilinx Inc., “System Generator for DSP Getting Started Guide”, Release 12.2, July 2010.

[17] Altera, “28-nm Variable-Precision DSP Block Architecture”, http://www.altera.com/technology/dsp/variable-precision/dsp-variable-precision.html, 2010.

[18] Altera, “DSP Builder”, http://www.altera.com/products/software/products/dsp/dsp-builder.html, 2010.

[19] http://proxacutor.free.fr/.

Chapter 5

Simulation of an OFDM Signal Transmitter and Receiver in Matlab

5.A) Introduction

On this chapter a partial implementation of the developed transmitter/receiver system is presented with similarities to the 802.11-2007 standard release. The purpose of the project was twofold: implement components of the OFDM signal transmitter/receiver and explore the capabilities of the XtremeDSP Development Kit-IV as well as the System Generator for DSP. The implemented blocks for the transmitter are: Data Source; 16-QAM Mapping; Pilot tones/DC subcarrier insertion; *Inverse Fast Fourier Transform* (IFFT), which also adds the Cyclic Prefix; IFFT Synchronizer and Guard Interval Inserter and Frame Assembly. On the receiver side there is an Input Buffer; CP Removal, *Fast Fourier Transform* (FFT), Pilot tones/DC subcarrier removal and 16-QAM Demapping. The following subchapters present a description of each main block on the implementation. A description of the XtremeDSP Development Kit-IV is performed on Annex A.

The system's operating frequency was set taking into consideration the ADCs sampling rate (105 MSPS) and the IFFT/FFT operation, which takes 3N clock cycles to process a block of N samples due to the algorithm that is used to calculate it. In order to achieve the channel bandwidth of 20 MHz set by the standard, the system must have an operating frequency of 100 MHz.

5.B) Transmitter

Figure 5.1 shows us the block diagram of the transmitter developed during the project.

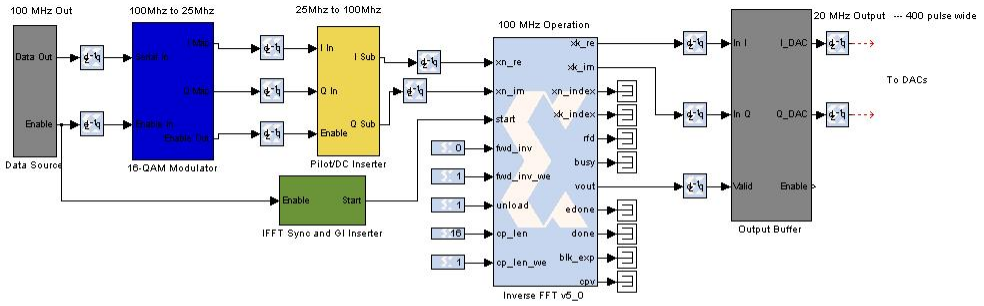


Figure 5.1: Block diagram of the OFDM transmitter

1. Data Source

It is the purpose of the block illustrated on Figure 5.2 to generate random data in order to be modulated according to the defined system and then demodulated at the receiver. There are 3 factors that are important to its assembly, the block's frequency output, the bit amount per stream and the spacing between each stream.

The spacing between streams is dependent on the time taken by blocks ahead on performing their operation. The only relevant block in this case is the IFFT, which lasts about 3*N clock cycles to process a block consisting of N samples.

The number of data bits per OFDM symbol depends on the type of modulation used. Taking into account that 16-QAM was chosen and that each OFDM symbol has 48 data subcarriers, a total of 192 bits per stream is ideal to avoid further processing, given that each 16-QAM symbol uses 4 bits (48*4=192).

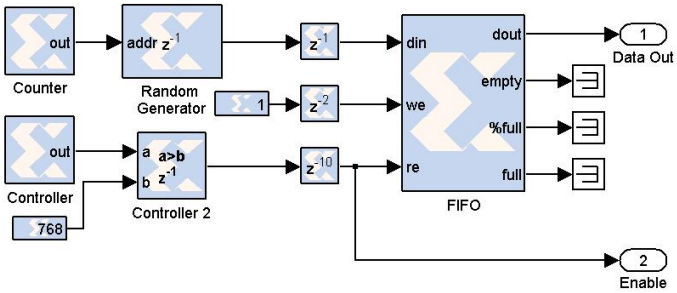


Figure 5.2: Data Source system

The counter performs a single step count from 0 to 190 with an explicit period of 5. If the system runs at 100 MHz, the data is generated at 20 MHz and enters the FIFO at the same rate.

The *Write Enable* signal is active all the time, whereas the *Read Enable* signal is only be active 20% of the time. The controllers attached to the *Read Enable* control the spacing between each stream, as well as the number of bits released in each stream. With the first controller constantly running from 1 to 960, the *Read Enable* signal will activate each time the counter goes above 768 (960 – 768 = 192 bits) giving each frame a spacing of 4*N between each other with each frame having N bits. Since the initial controller is specified to have an explicit period of 1 the data stream leaves FIFO with a rate of 100 MHz.

Figure 5.3 shows the data burst generated by the ROM on top and the stream of data ready to enter the 16-QAM Modulator block is shown below.

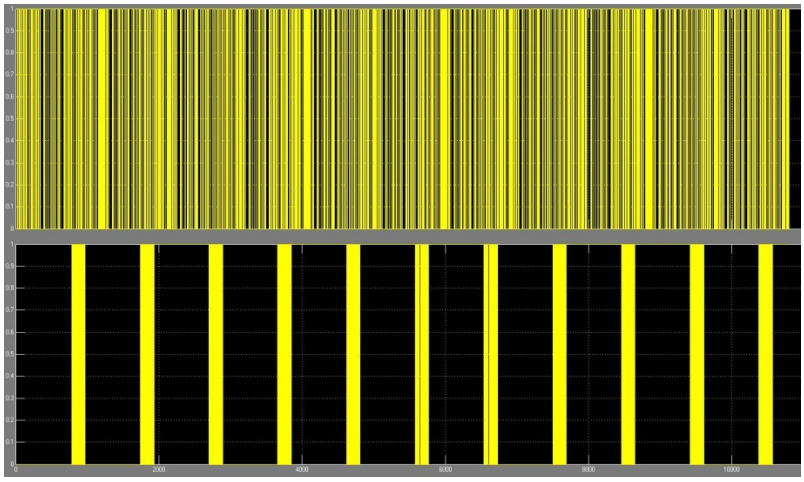


Figure 5.3: Data burst created by the Random Generator (top) and FIFO output (bottom)

2. 16-QAM Modulator

Following the FIFO's output on the Data Source the bits are mapped according to the 16-QAM modulation technique as shown on Figure 5.4. The first block is the *Serial to Parallel* block that generates 4-bit, which are then sliced into 2-bit words. Each word addresses a ROM to generate the constellation respective vector value according to Figure 5.5. The normalization factor is also applied by the ROM, $1/\sqrt{10}$ for the 16-QAM modulation. So for instance, the initial value vector parameter for *ROM I* is [-3 -1 +3

+1]/sqrt(10). The ROMs' output precision is a 16 bit signed signal, where 15 of them are binary.

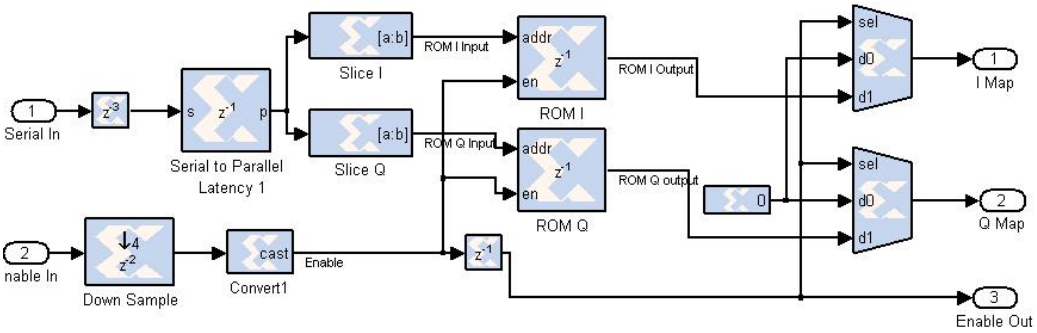


Figure 5.4: 16-QAM mapping system

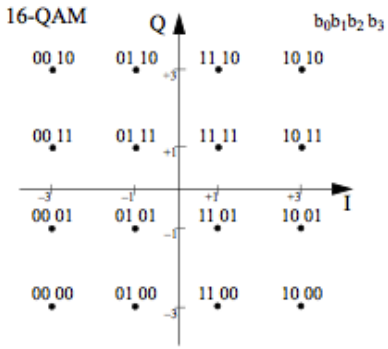


Figure 5.5: 16-QAM constellation bit encoding [1]

The data stream to be mapped isn't continuous at this block's input; therefore it becomes necessary to apply an *Enable* signal at the ROM's input in order to differentiate between a signal's absence and the all-zeros 2-bit word. The *Serial to Parallel* block requires 4 clock cycles for each word, which reduces the output rate from a 100 to 25 MHz. The *Enable* signal is provided by the *Read Enable* signal on the Data Source block, which has to be downsampled by a factor of 4 in order match the symbol rate at the ROM's output. The decision on whether the ROMs' output is a signal or the lack of one is performed by the *MUX* system.

Any of the other M-QAM modulation methods can be applied on this system as long as each block's parameters on this system and on the 16-QAM Demapping system are altered to match the desired execution. For instance, on a 64-QAM system the *Down Sample* block would see its sampling rate changed to 6.

Figure 5.6 shows an illustration of one of the signals of the 16-QAM output.

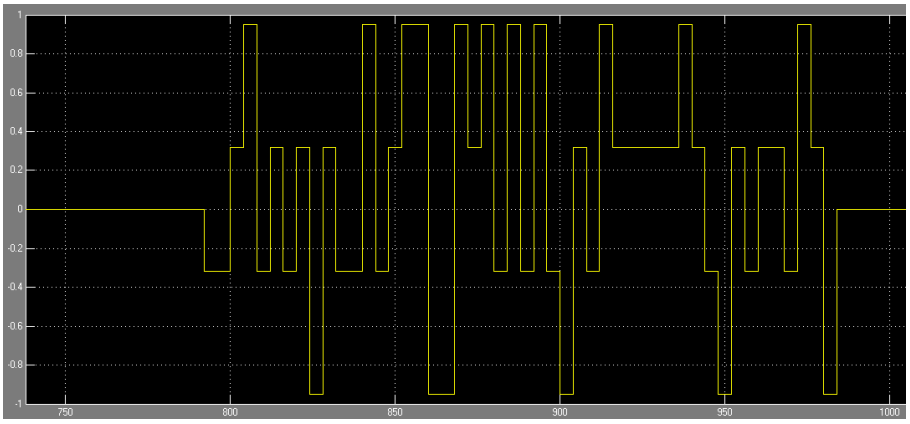


Figure 5.6: 16-QAM mapped bits with a frequency of 25 MHz

3. Pilot/DC allocation

The next subsystem on the transmitter chain performs the insertion of the pilot tones and DC subcarrier into the OFDM symbol. This means adding 4 pilot tones and the DC subcarrier, in order to make the coherent detection robust against frequency offsets and phase noise [1]. The system built is shown on Figure 5.7.

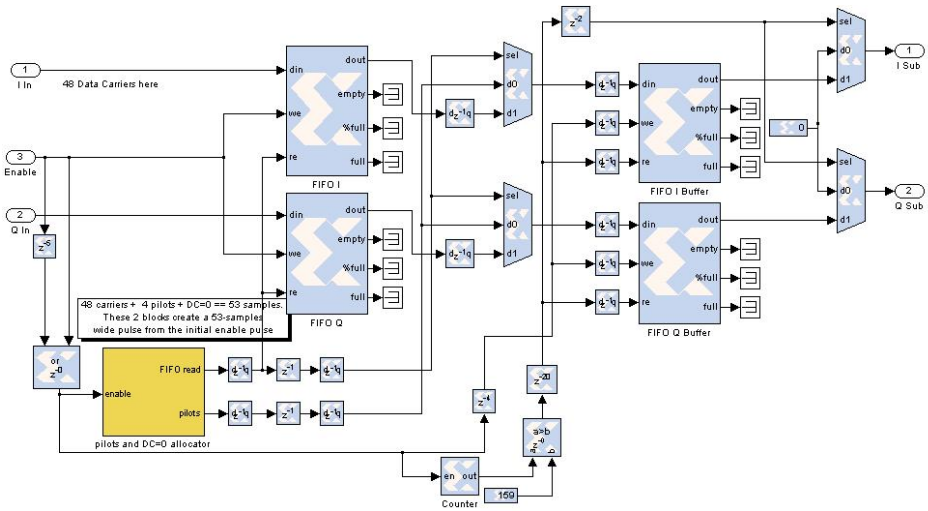


Figure 5.7: Pilot/DC inserter system

The first stage is composed by the 2 *FIFO's I/Q, pilots and DC=0 allocator* block and the following 2 *MUXs*. The FIFOs' task is to determine where both pilot tones and DC subcarrier are going to be inserted. This operation is performed by the *Read Enable* signal directly linked to the *pilots and DC=0 allocator* block shown on Figure 5.8. When the bit index (-26 to 26) on the 802.11 frame relates to a pilot tone or the DC subcarrier (-21, -7, 0, 7, 21) the FIFO stops the reading process from the data stored on it and releases a 0 instead for one of these 5 indexes. The signal's output for these FIFOs is a 53-bit frame running at 25 MHz containing 48 data subcarriers and 5 null chips. The value for the pilot tones and DC subcarrier is set to -1, 1 or 0 by the MUXs following the initial 2 FIFOs depending on their index type (pilot tone or DC).

The pilot tones originate from the *pilots and DC=0 allocator* block and their polarity is not variable on the implementation. In the 802.11-2007 standard release, the pilot tones polarities are controlled by a PN sequence that can be generated by the initial scrambler, but this subsystem is not presented on the work.

As mentioned in the previous subchapter (**2. 16-QAM Modulator**), the symbols to modulate have a symbol rate inferior to the one initially set. The *FIFO I/Q Buffers* compose the second stage, which has the purpose of restoring the symbol rate back to 100 MHz. The signal is written on the FIFOs at 25 MHz and the *Read Enable* signal is set to have an explicit period of 1, thus 100 MHz. The MUX blocks that follow the second stage have the same objectives has the ones on the 16-QAM Mapping system. On Figure 5.9, an illustration is shown of a 53-subcarrier symbol containing the 48 data subcarriers, 4 pilot tones and the DC subcarrier.

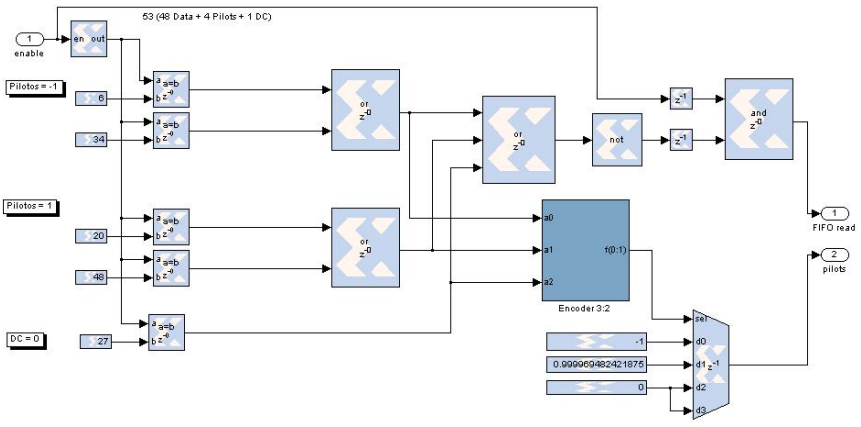


Figure 5.8: Pilot/DC allocator subsystem

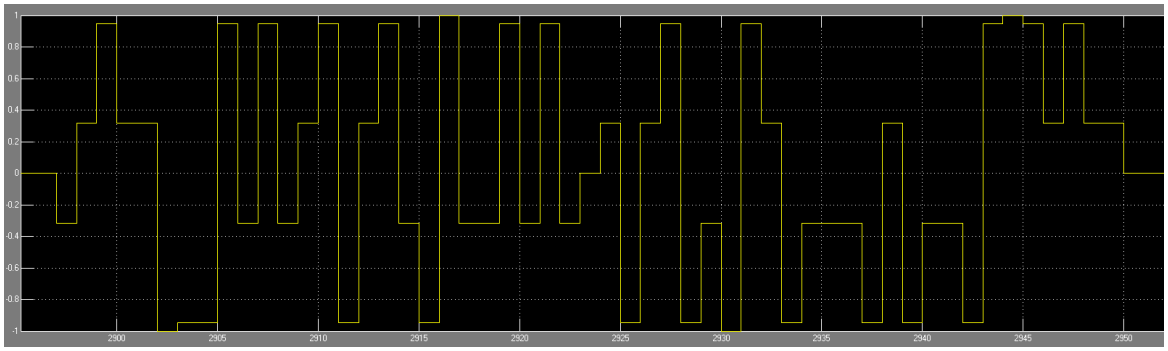


Figure 5.9: 16-QAM Mapped bits with 4 pilot tones and the DC subcarrier at 100 MHz

4. *Inverse Fast Fourier Transform* (IFFT)

The IFFT is one of the most important blocks when it comes to generating an OFDM signal because it guarantees the orthogonality between subcarriers. IFFT blocks are based on the *IP Cores* provided by Xilinx .The IFFT uses the *Radix-4 Burst I/O* implementation in order apply the Cooley-Tukey algorithm, which is considered to be the most appropriate for OFDM-based systems [2]. The other 3 possible implementations are Radix-2 Burst I/O, Pipelined Streaming I/O and Radix-2 Lite Burst I/O [3].

According to the standard, both IFFT data inputs must have a size of 64. The previous blocks have already defined 53 subcarriers and only the lower and higher frequency guard intervals remain to be set. Another of the IFFT inputs is the *start* signal, which when active starts loading the data and calculating the transform. Figure 5.10 shows the *IFFT sync and GI inserter* block. It processes the *Read Enable* signal (192-wide pulse) from the Data Source block making it narrower by creating a 64-wide pulse. The following delay block (z^{-65}) shifts the pulse to the appropriate position taking into account all the latency introduced by the 16-QAM Mapping and Pilot/DC inserter subsystems, leaving the signal active 6 samples before the 53-subcarrier signal and 5 samples after it.

Figure 5.10: IFFT synchronization and GI inserter

One of the features that the Fast Fourier Transform v5.0 has is the possibility to add the desired Cyclic Prefix length after the IFFT is performed without the need of any

additional blocks, thus reducing the system's complexity. The CP length is defined by the *cp_len* input on the IFFT representing the number of samples to be added, thus 16.

Figure 5.11 shows a representation of the serial signals after the IFFT is performed along with the added CP (red square), making a total of 80 samples (64 +16). It is clear on the figure that the 16 initial samples are a copy of the last 16.

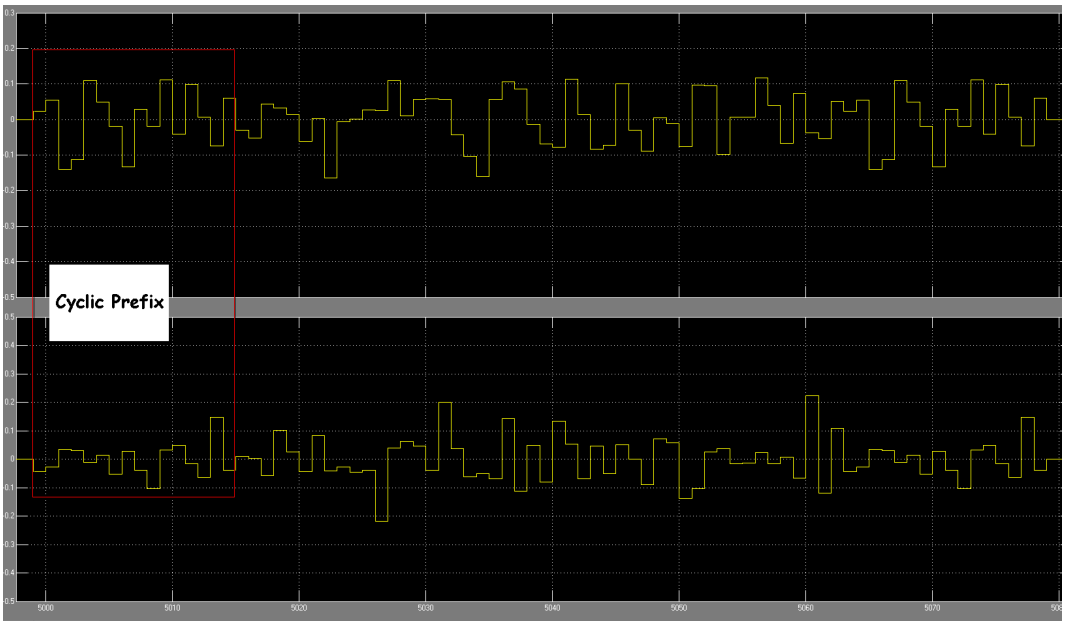


Figure 5.11: IFFT output with cyclic prefix; I (top) and Q (bottom)

5. Frame Assembly

The last block on the transmitter's side is based on two stages and is shown on Figure 5.12.

As stated in before in chapter **3.H) OFDM on 802.11-2007** each 802.11-2007 frame can contain up to 170 OFDM symbols. For simulation purposes 5 OFDM symbols per frame were used. This assembly is performed by the first stage of this block, which is composed by the *FIFO I/Q Output Buffers* and the *Frame Determination* block. Each 80-wide sample signal created by the IFFT enters the FIFO and it's read at 100 MHz. The *Frame Determination* block (can be seen on Annex A) is composed by a couple of relational blocks that are set to trigger the FIFO's *Read Enable* signal once the number of samples stored on the FIFO reaches the desired 400 samples (80 samples * 5 = 400 samples). The signal is stopped once the FIFO has released 400 samples and the process repeats itself.

Similarly to the stages found on the Data Source and Pilot/DC Inserter blocks, the Output Buffer also has to perform a symbol rate change. In this case to meet the channel bandwidth specification defined by the standard. Therefore, the signal between transmitter and receiver must have a channel bandwidth of 20 MHz and since the IFFT operates at 100 MHz, this frequency must be decreased by a factor of 5. The *Frequency Sync* block processes both writing and reading functions on the *FIFOs I/Q Out Rate Buffers*. The *Write Enable* synchronization signal consists of an extension of the IFFT's output signal *vout* in order to meet the 400-sample wide pulse, as well as processing in order provide a bigger spacing between each frame. The *Read Enable* synchronization signal is similar to the previous one, but instead of widening the pulse by adding more chips, the widening is performed by lowering the rate from 100 to 20 MHz, while the gap between frames reduces due to such widening. The 2 MUX blocks that follow the rate change have the same purposes has the ones on the 16-QAM Mapping system.

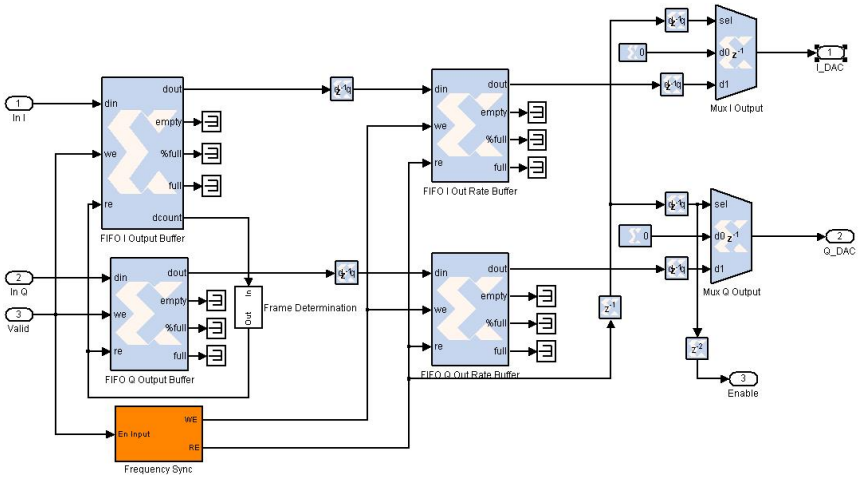


Figure 5.12: Frame Assembly system

Figure 5.13 shows signals that can be found on this block. On top the input is illustrated once the signals are processed by the IFFT. The middle signal shows the result of the concatenation of 5 OFDM signals into a single frame and bottom graph illustrates the signal sent to the DACs after the channel bandwidth is adjusted to match the standard specifications (20 MHz). The duration of a single frame at the transmitter's output is 20 µs, given that 80 samples times 5 signals divided by 20 MHz equals 20 µs.

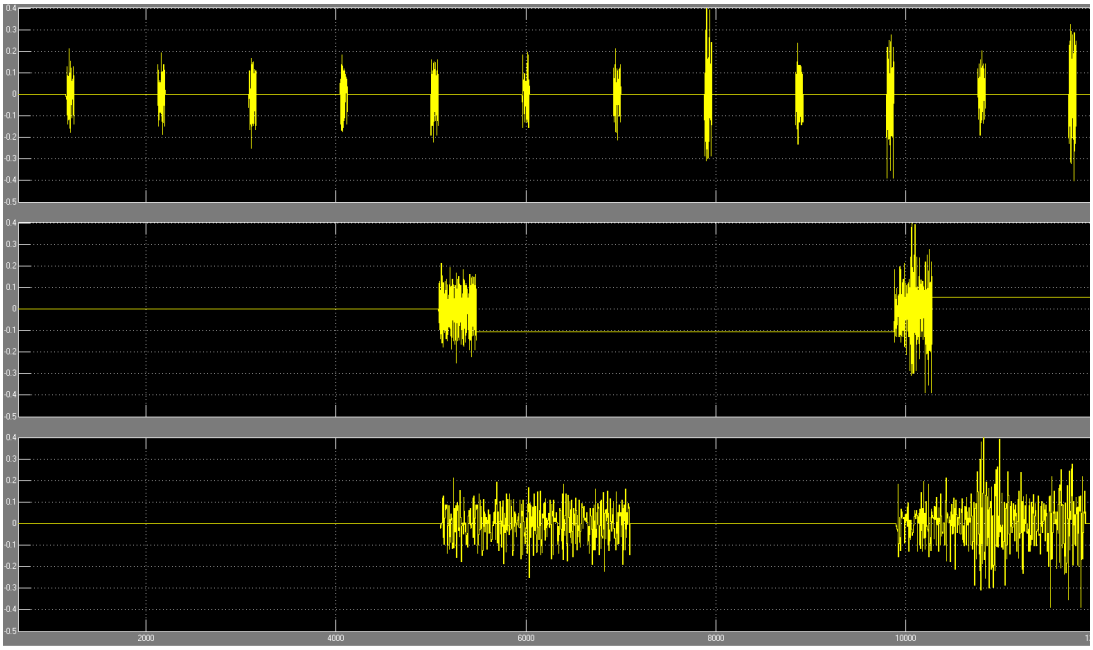


Figure 5.13: Output Buffer signals, Input (top) and 5 concatenated OFDM signals (middle) at 100 MHz, Transmitter's output at 20 MHz (bottom)

The remaining processing to be performed by a typical transmitter for this kind of signal once the Digital to Analog Conversion is performed would be an IQ modulator similar to the one shown on Figure 5.14. The opposite IQ demodulator would have to be presented on the receiver in order for the signal to be demodulated. These systems lie outside the scope of this project and therefore are not tested or presented.

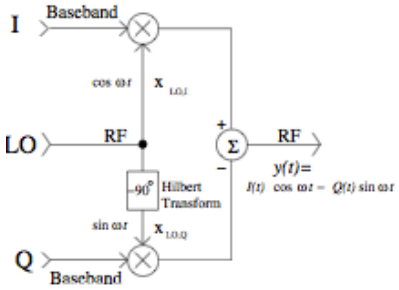


Figure 5.14: Ideal IQ modulator [4]

5.C) Receiver

The receiver implemented for this project was built in order to demodulate the signal created at the transmitter. Although in most wireless systems much of the complexity is found on the receiver side, the lack of channel impairments reduces some of the complexity that a typical receiver would show. Figure 5.15 illustrates the block diagram of the OFDM receiver.

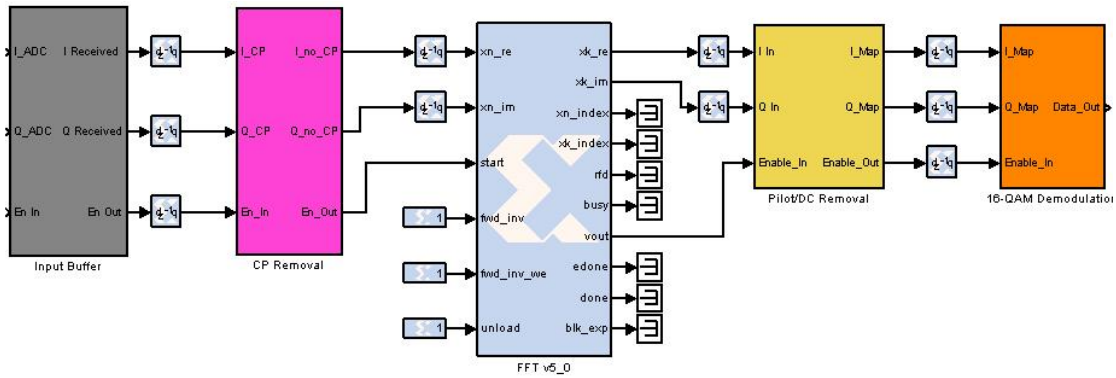


Figure 5.15: Block diagram of the OFDM receiver

1. Input Buffer

The first blocks that can be found on the receiving end of this system on Figure 5.16 are *Converter* blocks. The purpose of these blocks is to change the precision on the system's signal or its word type. Going back to subchapter **B)2: 16-QAM Modulator**, the signal precision set by the ROM was of 16 bits, where 15 of them are binary digits. The higher number of binary digits a signal has the higher its precision will be. This block was included to take into account if a simulation including DACs is performed given that a DAC changes the signal's precision to 14 bits with 13 binary digits. This particular precision is used due to the fact that the XtremeDSP Development Kit-IV possesses DACs with a 14-bit resolution.

The remaining Input Buffer blocks can be divided into to 2 stages similarly to the Transmitter's Frame Assembly block.

On the first stage the symbol rate is set back to 100 MHz. The stage operates in a similar way when compared to previous rate changes. The *Write Enable* signal is active the whole time while the *Re Sync* blocks consists of a counter with an explicit period of 1 in order to get the symbol rate back to 100 MHz.

The second phase has the role of splitting the frame back into single OFDM signals. The signal enters the FIFO freely with no restrictions or any kind of change on the *Write Enable* signal. The *Read Enable* signal is controlled by the FIFO's *empty* output. Once the FIFO reaches 80 samples it starts unloading them at the same rate, but with enough spacing between OFDM signals so that the FFT can be processed without interruptions Figure 5.17 shows the output of the input buffer block, a single OFDM signal.

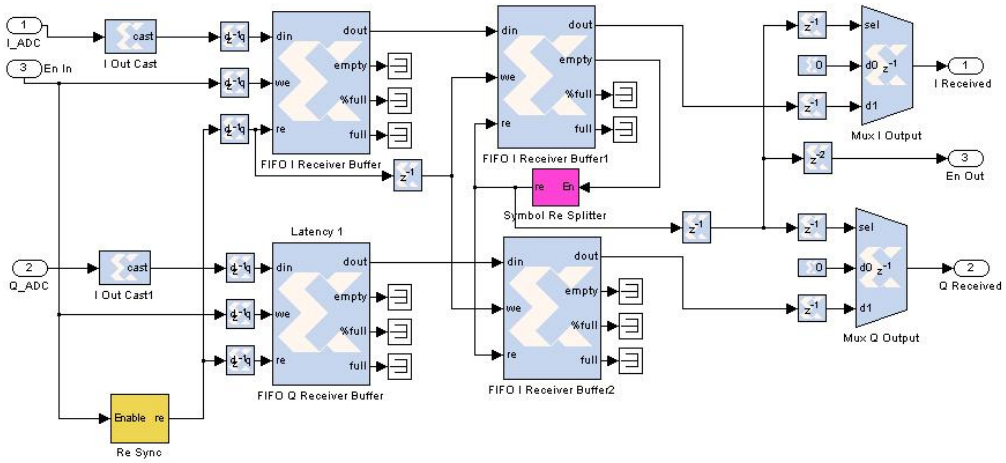


Figure 5.16: Input Buffer system

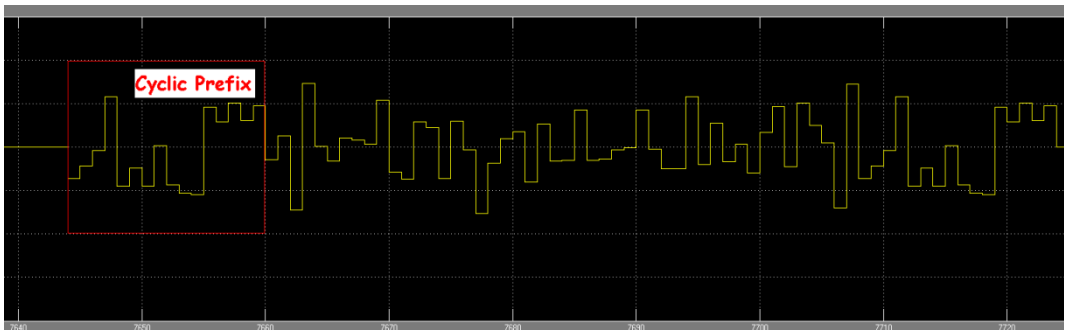


Figure 5.17: Single data stream at the input buffer's output at 100 MHz

2. CP Removal

Xilinx's FFT blocks only have the ability to add the CP, but not removing it. The CP removal is performed by the system shown on Figure 5.18.

The 80-wide pulse coming from the Input Buffer is narrowed with a *Delay* and an *And* block to a 64-wide pulse. The pulse is then applied to a MUX system; where the 16 first bits of each signal are zeroed. The 64-wide signal is shown on Figure 5.19.

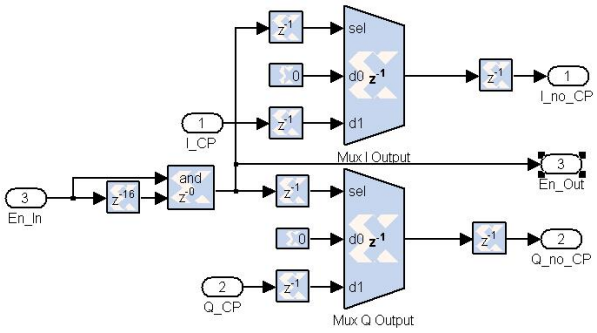


Figure 5.18: CP Removal system

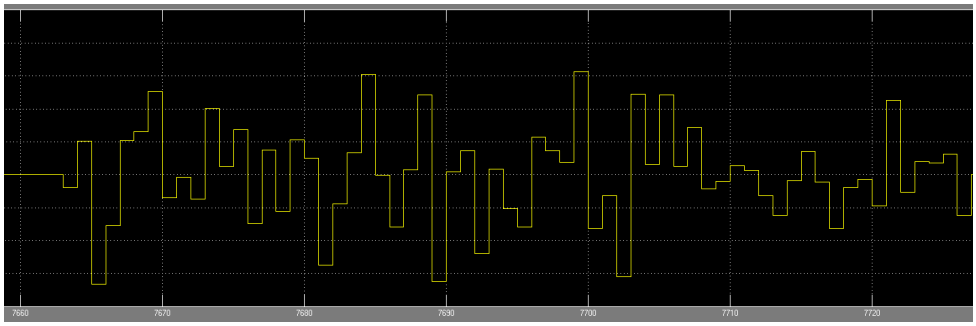


Figure 5.19: Received OFDM signal after CP removal at 100 MHz

3. Fast Fourier Transform (FFT)

Apart from being the complete opposite operation of the IFFT since both create a matched linear pair [5], the configuration required by the FFT is not that different when compared to the IFFT (subchapter **B)4. *Inverse Fast Fourier* Transform (IFFT)**).

The block's datasheet [3] determines that the *start* signal must be active 4 time samples before the I/Q signals reach their respective inputs. The FFT processing of a single OFDM signal is shown on Figure 5.20.

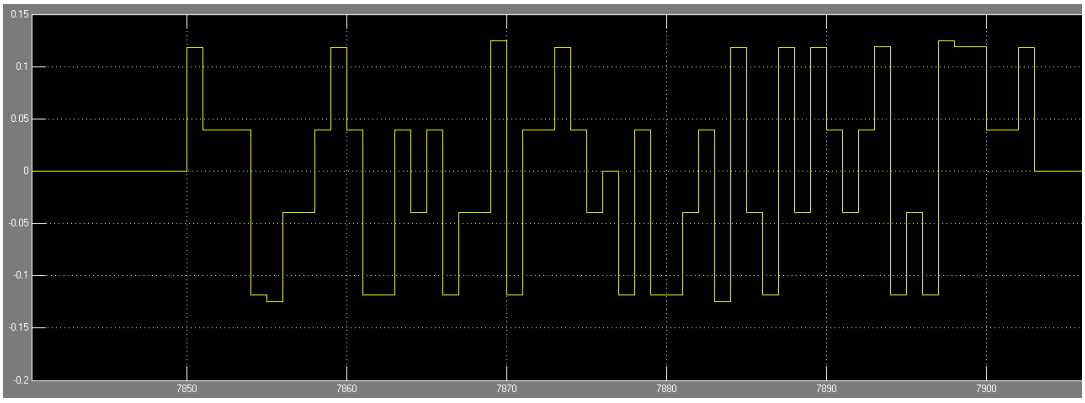


Figure 5.20: FFT Output at 100 MHz

4. Pilot/DC Removal

This block's purpose is the removal of the 4 pilot tones and the DC subcarrier introduced on the transmitter on each OFDM signal.

On the first stage, the *pilots and DC=0 allocator* subsystem's purpose is to identify the indexes of where the pilot tones and the DC subcarrier are located on the signal. Once identified the FIFO won't write the bit, proceeding to the next symbol. The *Read Enable* signal is shortened from the 53-wide pulse that leaves the FFT to a 48-wide pulse, matching the number of data subcarriers.

On the second stage, the symbol rate is lowered in a similar way to previous symbol rate changes so that the rate at the following subsystem output is equal to the one presented at the Data Source's output. The *Read Enable Sync* subsystem purpose is to lower the symbol rate, downsampling it by a factor of 4 so that the signal is read at a rate of 25 MHz. Figure 5.22 illustrates an OFDM signal at this subsystem's output containing the 48 data subcarriers without any pilot tones or the DC subcarrier. An incoherence can be found after time slot 8050 due to the precision presented by the system along with the FFT operation, but it won't affect the receiver's output given that the final signal is presented with a precision of a single bit.

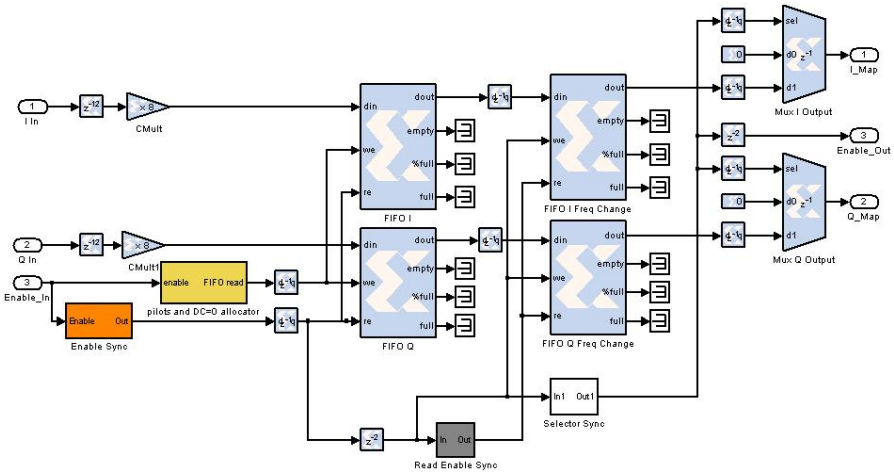


Figure 5.21: Pilot/DC Removal

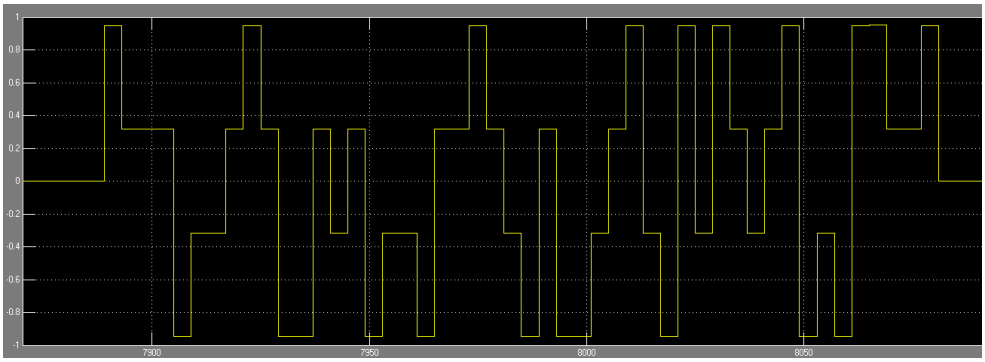


Figure 5.22: Single OFDM signal after FFT processing and Pilot/DC Removal

5. 16-QAM Demapping

Since the data was normalized when the mapping was performed at the transmitter, the first step taken on this block is the de-normalization of the signal by multiplying both signals by $\sqrt{10}$ (Figure 5.23 yellow block).

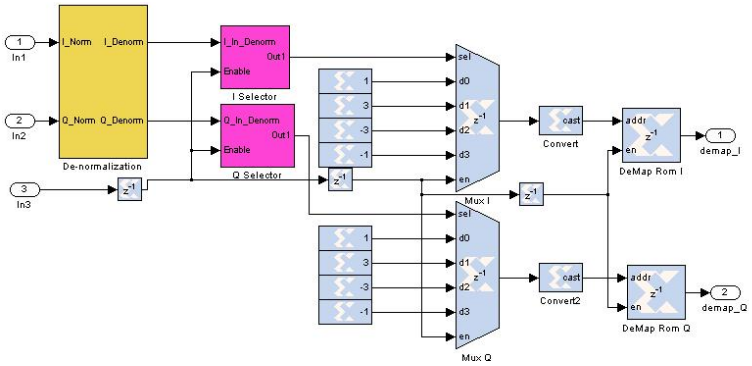


Figure 5.23: De-mapping subsystem

Once the symbols are de-normalized, quantification is performed to decide where they are located on each of their axis. Quantification is performed with the conversion of the 16-bit signals to 2-bit signals, thus creating decision thresholds. For example, if an I signal symbol has a value of 00, by looking at the 16-QAM constellation on Figure 5.5, the I signal has a value of -3.

After locating each symbol in the axis, the respective coefficient is attributed by the MUX. Once the coefficients are attributed, the I and Q signals are de-mapped by the ROM into a 0's and 1's signal so that they can concatenated into a 4-bit word. The process finishes with a downsampling of the signal by 4 so that the symbol rate at the *Parallel to Serial Converter* (Figure 5.25) matches the 100 MHz at the Data Source's.

Figure 5.26 shows a comparison between the signal at the Data Source's output on the transmitter and the one on the 16-QAM Demodulator at the receiver.

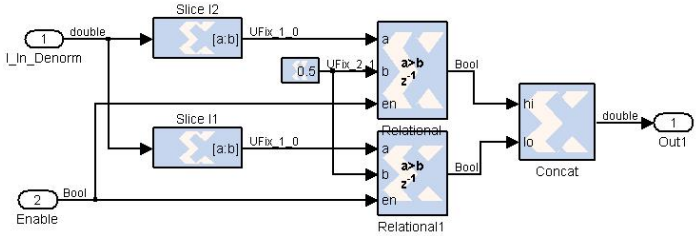


Figure 5.24: I/Q Selector subsystem

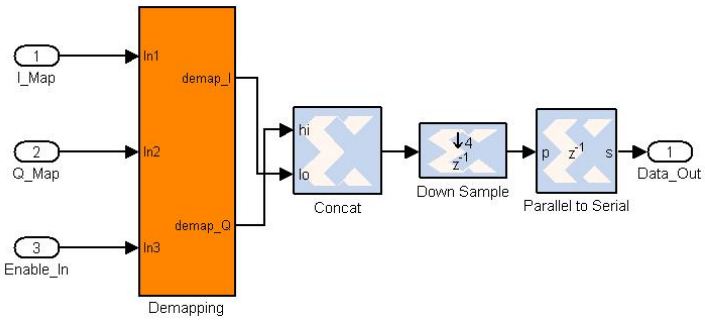


Figure 5.25: 16-QAM Demodulator

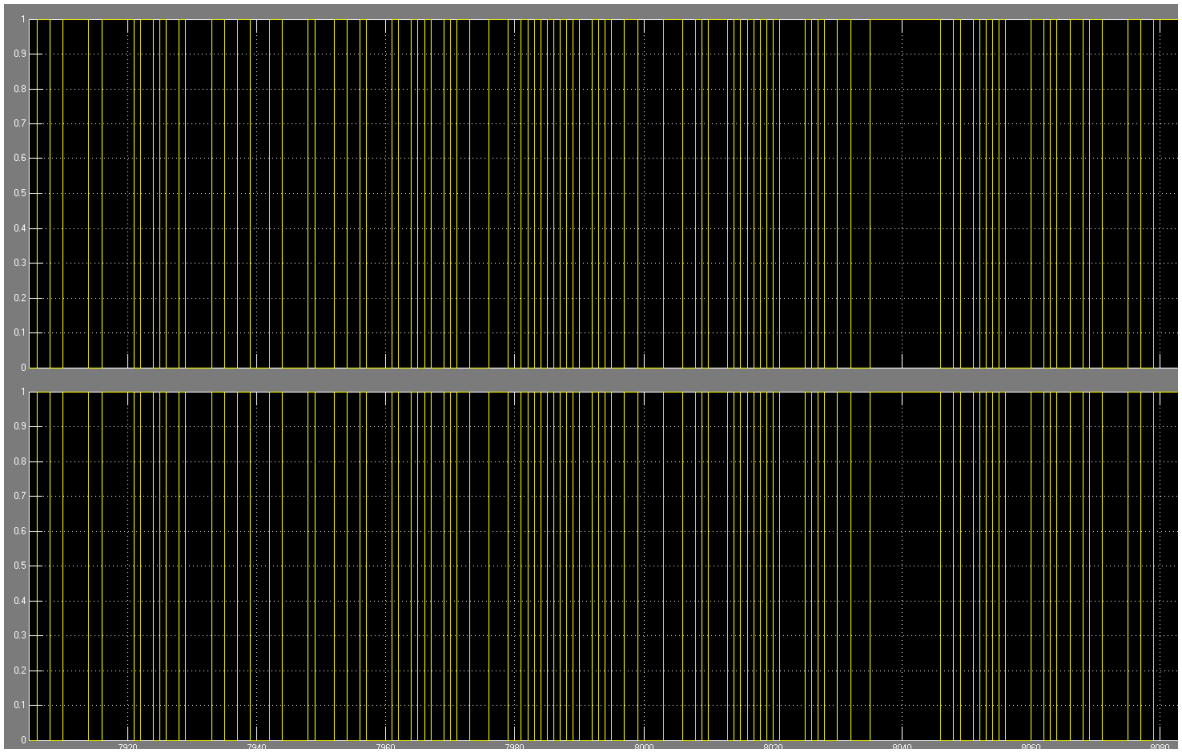


Figure 5.26: Data Source block output (top) and 16-QAM Demodulation block output (below)

Bibliography

[1] IEEE Computer Society, "IEEE Standard for Information technology—Telecommunications and information exchange between systems—Local and metropolitan area networks— Specific requirements – Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications", IEEE Std 802.11™ – 2007, 12 June 2007.

[2] J. Tian, Y. Xu, H. Jiang, H. Luo & W. Song, "Efficient Algorithms of FFT Butterfly for OFDM Systems", Proc. IEEE 6th CAS Symposium On Emerging Technologies: Mobile and Wireless Communications, pp. 461-464, Shanghai, China, June 2004.

[3] Xilinx Inc. "Fast Fourier Transform v5.0", October 2007.

[4] Shashank Mutha, et al, "Technique for Joint Balancing of IQ Modulator-Demodulator Chains in Wireless Transmitters", IEE MTT-S International, pp 221-224, June 2009.

[5] "Orthogonal Frequency Division Multiplex", Tutorials in Communications Engineering, http://www.complextoreal.com/tutorial.htm, 2010.

Chapter 6

FPGA Co-Simulation of an OFDM Signal Transmitter

6.A) Introduction

This chapter presents the FPGA Co-Simulation of the transmitter that was developed on the previous chapter.

Hardware Co-Simulation is a methodology introduced by Xilinx on 2003 [1], which allows a system simulation to be run partially in Simulink and partially in hardware (FPGA). This enables accurate hardware modeling along with faster simulation times for the parts implemented in hardware due to the faster calculations and easier hardware verification by implementing the manufactured algorithm into the FPGA.

Xilinx's block components behaviors are projected to Simulink, while at the same time, the behavior of each block's associated hardware component is performed on the FPGA. The objective is to get both hardware and software working before the prototyping stage by providing a better understanding of its behavior.

6.B) Advantages and Disadvantages

There are several advantages provided by Hardware Co-Simulation. Previous knowledge of HDL is not longer a requirement in order to design a system; SysGen compiles the design into a library block in order to bring the FPGA to the Simulink environment and the same design is used for the co-simulation. The full understanding of how the kit that hosts the FPGA works also isn't mandatory.

There are however some disadvantages associated with the methodology. Although the speed increase in simulation is outstanding, compiling a library block is a lengthy process and unlike simulating solely on Simulink, the change of a parameter on a Sysgen block or even a link between them demands the re-compilation of the associated

library block, a process that can take up to more than 15 minutes depending on the amount of total resources used by the FPGA and the CPU in use. Another is that the design is totally visible to us when simulating in Matlab, unlike what is implemented on the FPGA.

6.C) Implementation

The structure implemented for Hardware Co-Simulation is in many ways similar to the one implemented on Matlab. Small differences are only related to the output precision defined by the DACs. The implemented transmitter is show on Figure 6.1.

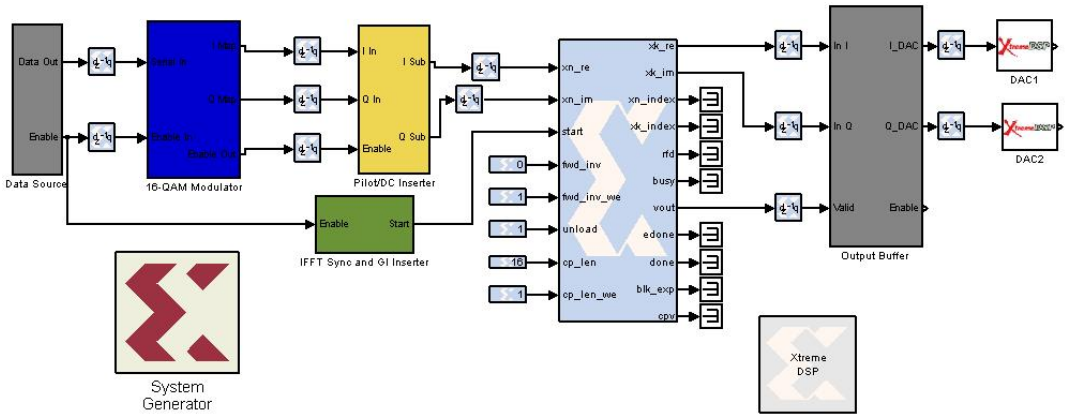


Figure 6.1: Transmitter Setup for Hardware Co-Simulation

The steps taken to create an XtremeDSP by altering the *System Generator* block parameters are the following:

1. Set Compilation target, XtremeDSP Development Kit (PCI)
2. Set Compilation part, Virtex4 xc4vsx35-10ff668
3. Set FPGA clock period, this limits the free running clock parameter on the library block. When using DACs/ADCs a free running clock is mandatory.
4. Generate the library block

Once the compilation is performed, the block is created and a new window appears with the block. The block must then be dragged into the compiled system environment and the simulation is ready to run. Every block apart from the library block

and the *System Generator* block can be erased from the environment if needed. As mentioned above, the *Clock Source* parameter on the library block must be changed to *Free running*.

6.D) Results

Figure 6.2 shows the output of a partial 16-QAM signal with an operating system's frequency of 100 MHz. At the 16-QAM Mapping system each symbol has a total duration of 1.92 μs has predicted, given that the signal leaves the 16-QAM modulator with a rate of 25 MHz. Therefore each symbol has a duration of 40 ns (1/25e6) and each 16-QAM signal has a total duration of 1.92 μs (=48/25e6). The high sample rate in use doesn't allow the DAC to calculate the signal properly causing inaccuracies on it, although the symbols can still be detected. Figure 6.3 illustrates 2 frames at the transmitter's output with each frame having a total duration of 20 μs as expected.

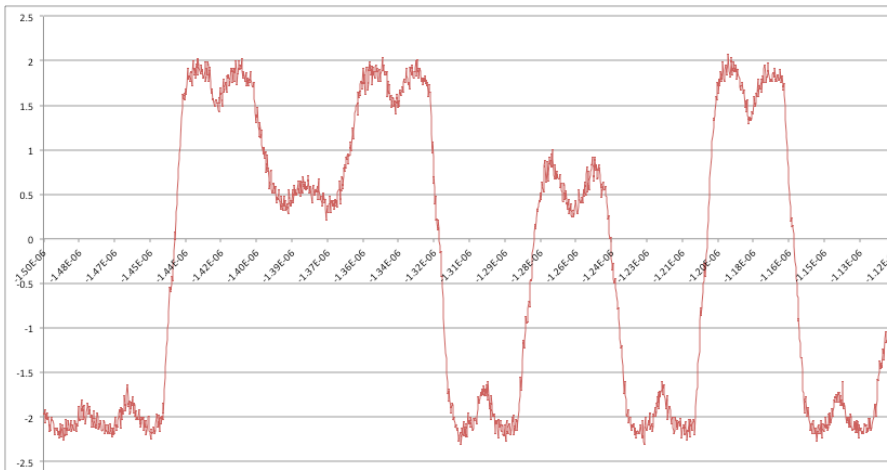


Figure 6.2: Nine 16-QAM Symbols at 25 MHz

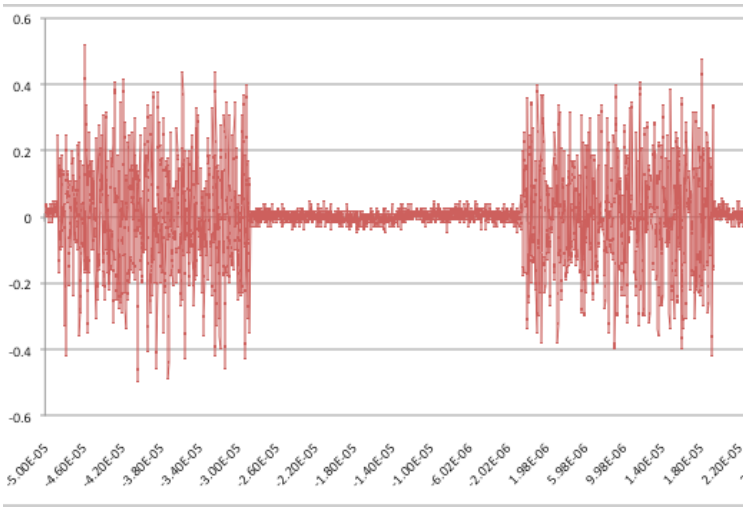


Figure 6.3: Full Transmitter DAC's output at 20 MHz

6.E) Resources

When the library block for the FPGA co-simulation is performed, the map, place and route process takes place providing the FPGA resources that are used. Table 6.1 presents an estimated of the used resources that were required for the transmitter's implementation along with the total percentage used from the Virtex-4 FPGA.

Slices	Flip-Flops	Block Rams	LUTs	IOBs	Embedded 18x18 Multipliers
2530 (16%)	3551 (11%)	19 (9%)	3549 (11%)	39 (8%)	9

Table 6.1: FPGA resources used for the OFDM Transmitter implementation

Bibliography

[1] Nabeel Shirazi and Jonathan Ballagh, “Put Hardware in the Loop with Xilinx System Generator for DSP”, Xcell Journal, Fall 2003.

Chapter 7

Conclusions and Future Work

7.A) Conclusions

The main objective of this project was the development of an OFDM signal transmitter and receiver using the Xilinx's System Generator for DSP environment.

Before the implementation began, the use of other platforms for digital signal processing for communications was performed showing not only the FPGA internal architecture, but also the advantages brought by the platform to such work when compared to two different types of platforms. Xilinx's CAD environment System Generator for DSP's potential was studied given that it was the main tool for this work, showing the easiness of work provided not only by Xilinx's software. Avoiding a full implementation of an OFDM signal using VHDL is to avoid thousands of code lines as well as the compilation time lost for error tracking or system testing. The potential brought by the 802.11n amendment will be the future of most wireless environments and soon enough it will bring the downfall of any wired enterprise workplace with many companies certainly replacing cables for antennas.

Matlab's simulation proved to be very similar when compared to a hardware simulation, although the software's version is already out of term and even though a block diagram will hardly provide any insight on a hardware implementation regarding its display. Even so, the hardware co-simulation mode makes unnecessary any knowledge that would be needed on the clock routing/controlling for the Virtex-II board that controls the clocks. Knowledge on VHDL would also be needed if the FPGA simulation were to be run through bitstreams instead of the Simulink environment. The remaining System Generator for DSP blocks and methodologies also require some study and knowledge, but are in many ways similar to Simulink blocks making them easier to work on.

7.B) Future Work

The work developed on this project can be continued in several directions depending on the scope intended:

- Some OFDM system blocks weren't implemented, such as the ones related to coding and decoding, interleaving and deinterleaving.
- As well subsystems for filtering, PAPR treatment, LMS analyzer, PN sequence generator for the scrambler and pilot polarity, among others.
- An IQ modulator can be applied after the DAC's, or the opposite on the receiver in order to throw the signal to Radio Frequency.
- MIMO can also be implemented, although only at a Simulink/SysGen level due to the limited number of DACs.
- A different kind of data to be transmitted can also be generated. For example, replacing the Data Source block for a data processing block that turns an image into bits and can adequate its resolution.
- Due to the lack of switches and buttons on the XtremeDSP Development Kit it would be useful to create a board with an user-friendly interface to interact with the kit's pins.

Annex A

XtremeDSP Development Kit-IV

A.1 Introduction

The hardware implementation present on Chapter 6 was performed on the XtremeDSP Development Kit-IV show on Figure A.i. It was built by Nallatech and Xilinx and dedicated for the implementation of systems focused on digital signal processing. The platform interacts with Xilinx System Generator for DSP giving a smooth transition between top-level block algorithms to hardware synthesization. It also allows performing Hardware Co-Simulation on the Kit via a PCI or JTAG interface.

The following description is based on Nallatech's XtremeDSP Development Kit-IV user guide [1].

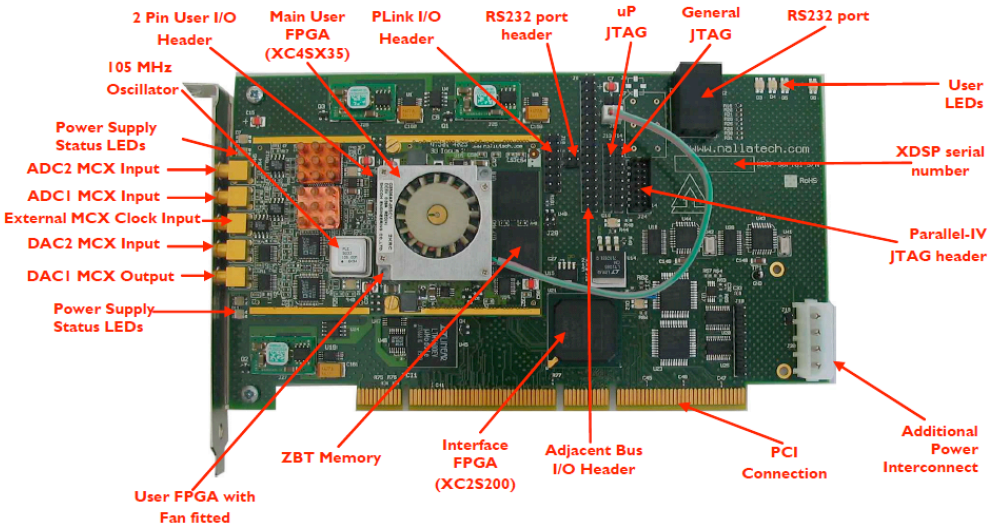


Figure A.i: XtremeDSP Development Kit-IV

The whole kit consists of the following main elements:

- User FPGA: Virtex-4 Model XC4VSX35-10FF668
- User Clock FPGA: Virter-2 Model XC2V80-4CS144

- Interface FPGA: Spartan-2
- 2 Independent ADC Channels Model AD6645 ADC
- 2 Independent DAC Channels Model AD9772 DAC
- PCI, JTAG or USB interfaces to the CPU

The kit consists of a motherboard which has a module inserted. The User Clock FPGA can't be seen on Figure A.i because it is located underneath the module. The motherboard is called BenONE-Kit and the module is referred to as BenADDA Dime-II. Of the elements referred to above, the module contains the User FPGA, User Clock FPGA, DACs and ADCs. Figure A.ii shows the kit's functional diagram.

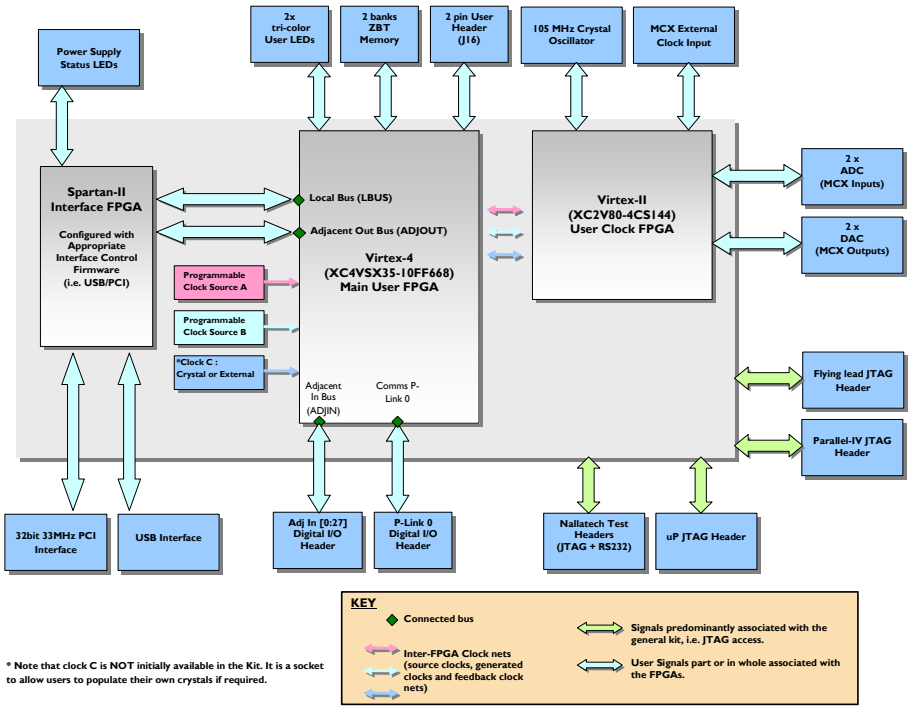


Figure A.ii: XtremeDSP Development Kit-IV Functional Diagram

A.2 FPGAs

The Virtex-4 FPGA is fully dedicated to user's design purposes and as shown on Table 4.1 contains over 34.500 logic cells, 192 XtremeDSP slices and reaches a maximum frequency of 500 MHz.

The Spartan-II FPGA controls the interface with the computer. The platform's possible interfaces are PCI, JTAG or USB. PCI was the interface used for the project. The

FPGA interacts with the User FPGA via a dedicated bus composed of the Local Bus (LBUS) and the Adjacent Out Bus (ADJOUT) connections.

The Virtex-II FPGA purpose is to control the clocks in a user design and on the DACs/ADCs.

A.3 DACs

The kit possesses two analog outputs implemented by two AD9772A DACs built by Analog Devices [2]. Clocks that are forwarded by the Virtex-II control both DACs independently. The DACs main features are a 14-bit resolution and a 160 MSPS maximum input data rate. Figure A.iii illustrates the interface between one of the DACs and the FPGAs.

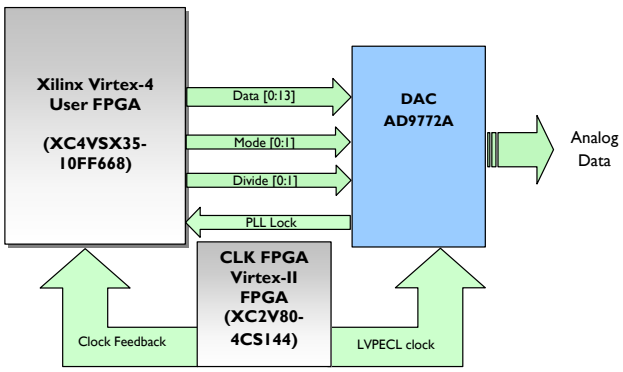


Figure A.iii: DAC Interface

A.4 ADCs

The kit possesses two analog inputs implemented by two AD6645 ADCs built by Analog Devices [3]. Clocks that are forwarded by the Virtex-II control and manage both DACs independently. The ADCs main features are a 14-bit resolution and a 105 MSPS sampling data rate. Figure A.iv illustrates the interface between one of the ADCs and the FPGAs.

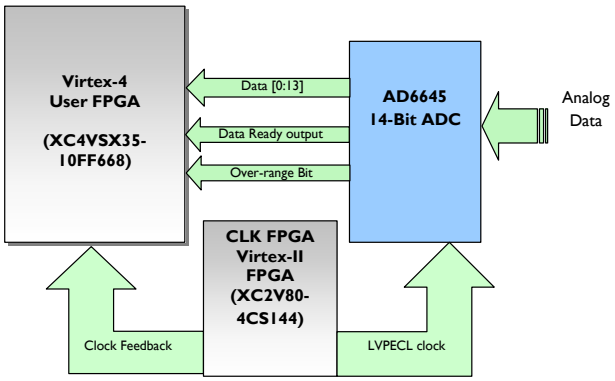
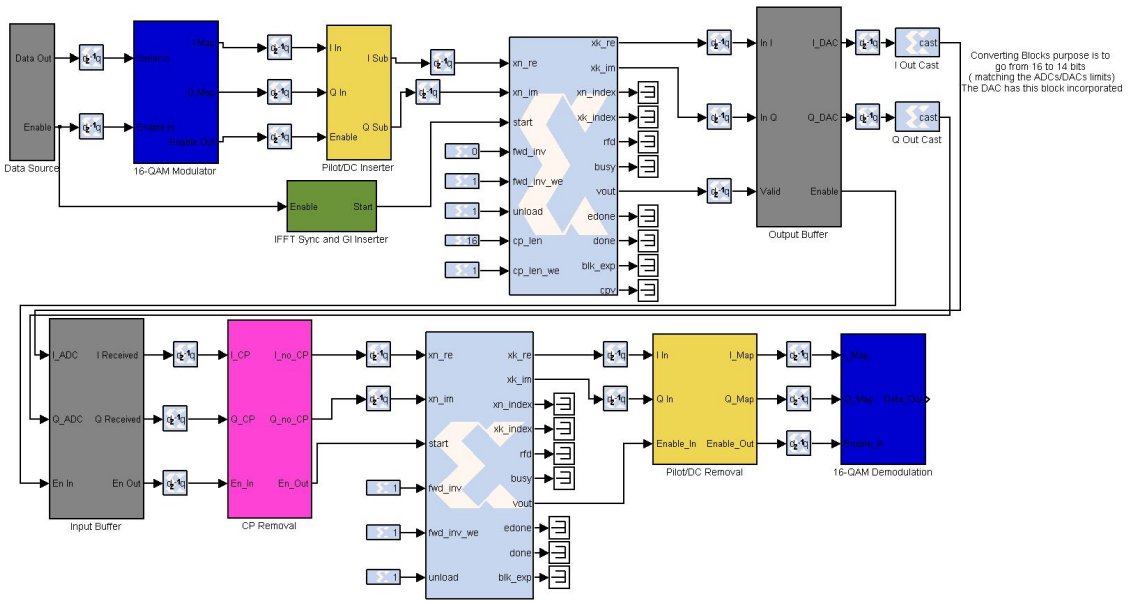


Figure A.iv: ADC Interface

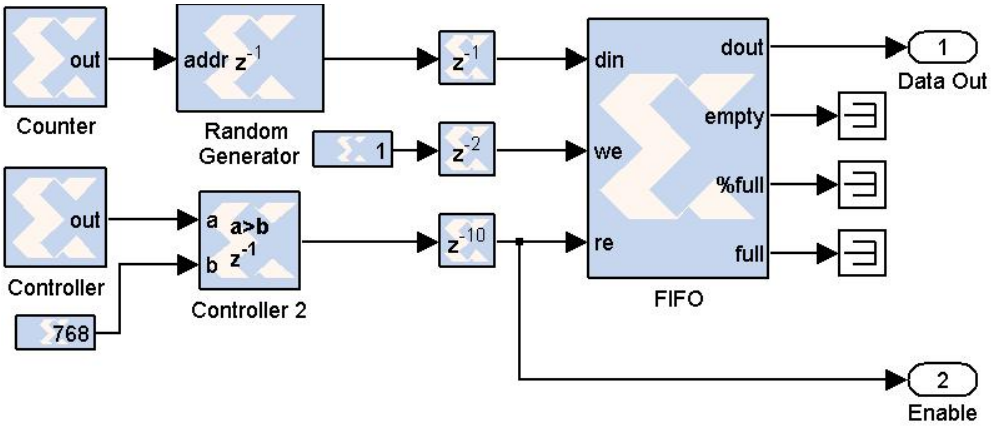
Annex B

Block diagrams of the developed system

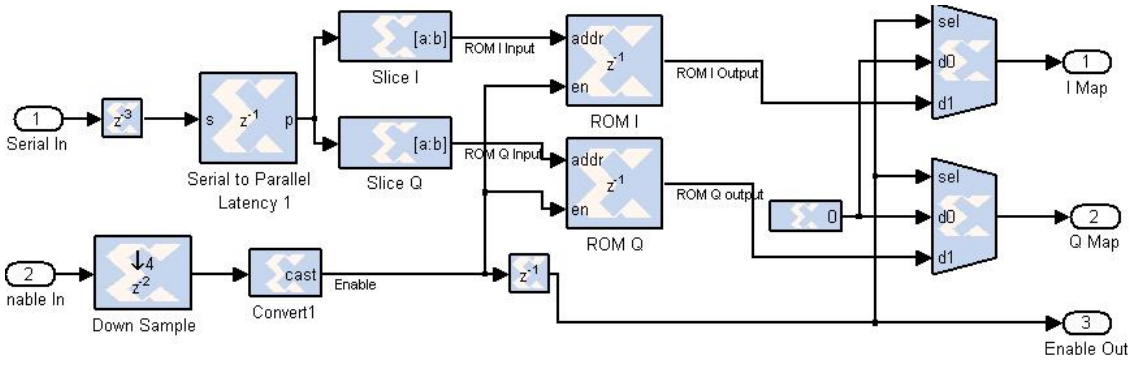
B.1. Full System



B.2. Data Source

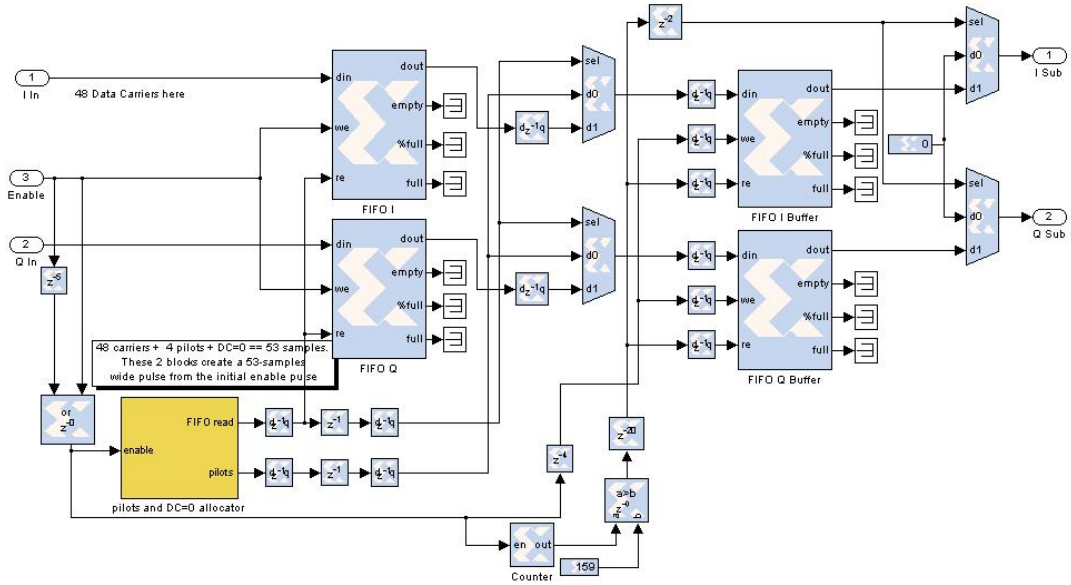


B.3. 16-QAM Modulator

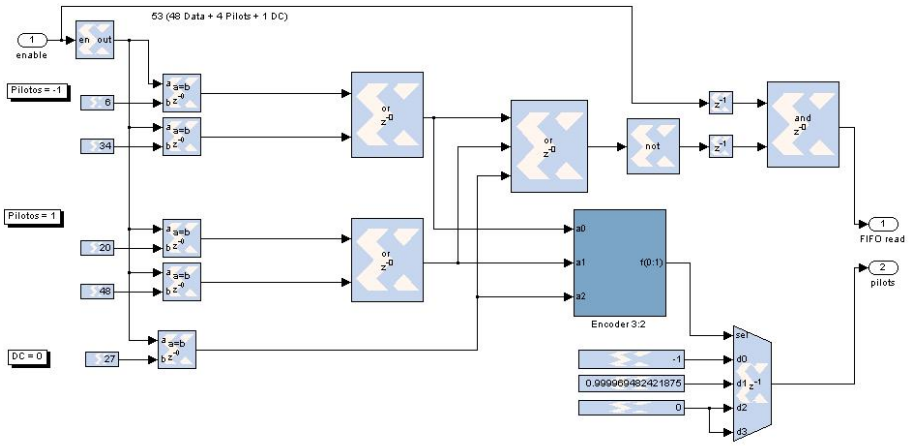


B.4. Pilot/DC Inserter

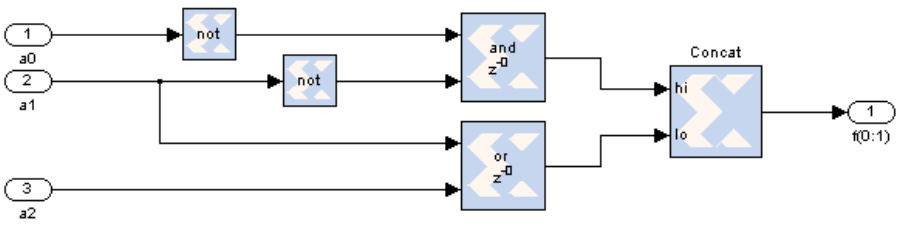
MAIN BLOCK



PILOTS AND DC=0 ALLOCATOR

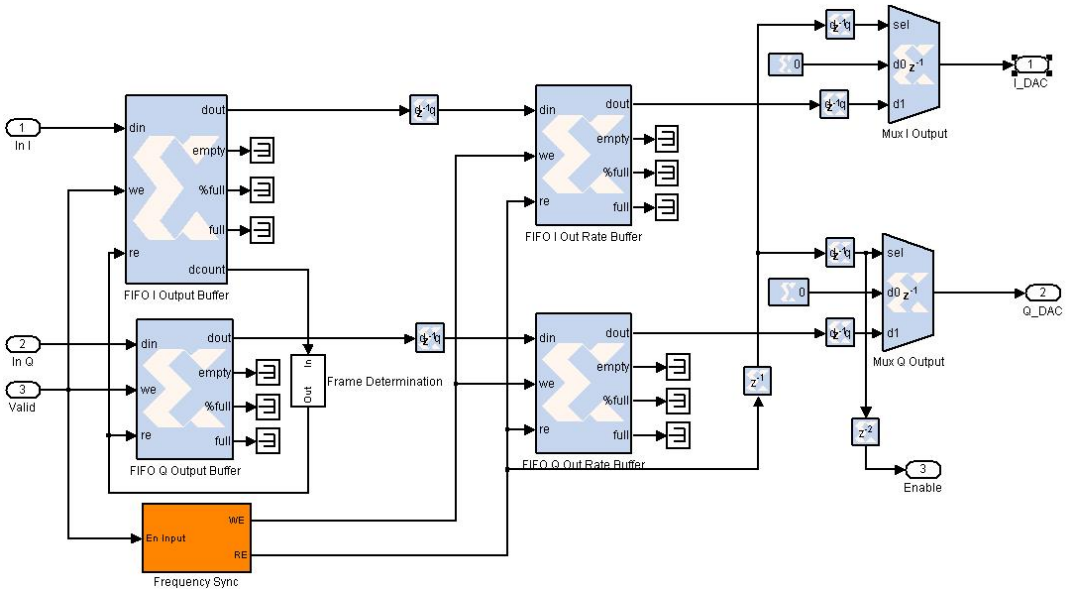


ENCODER 3:2

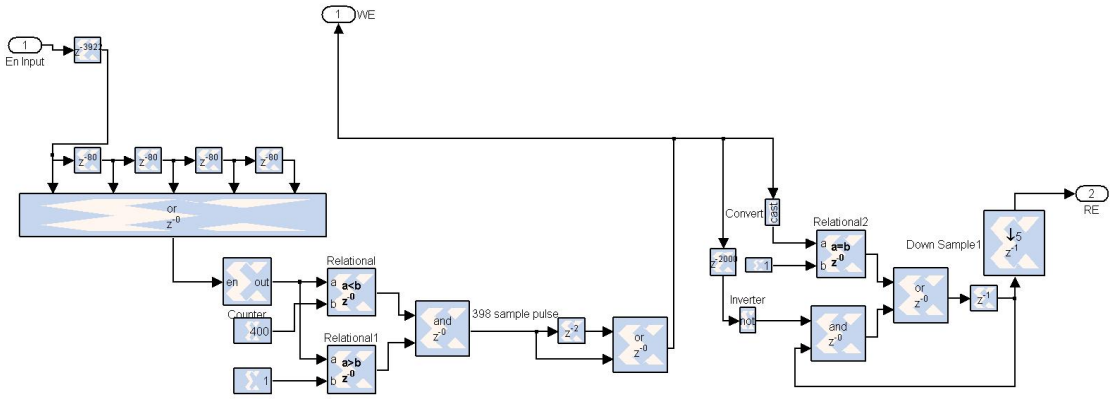


B.5. Frame Assembly

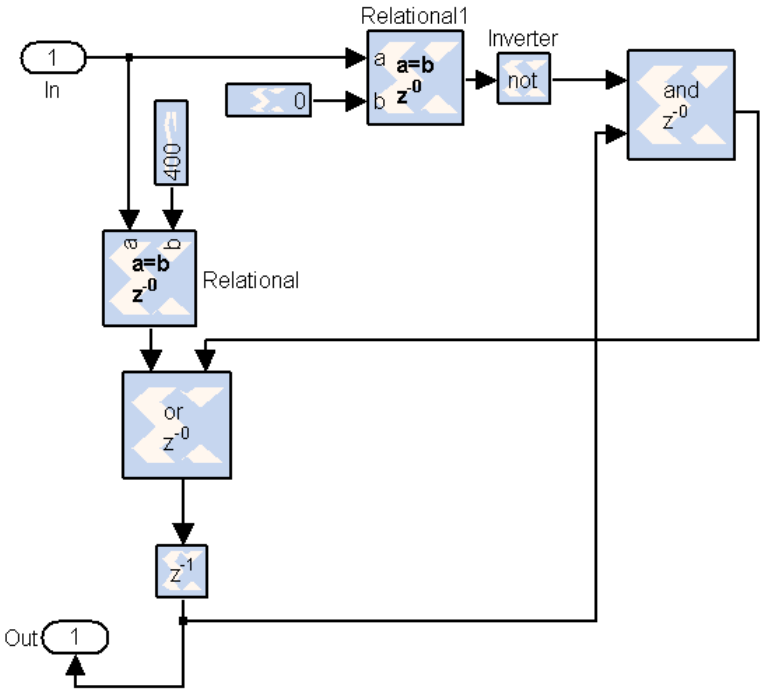
MAIN BLOCK



FREQUENCY SYNC BLOCK

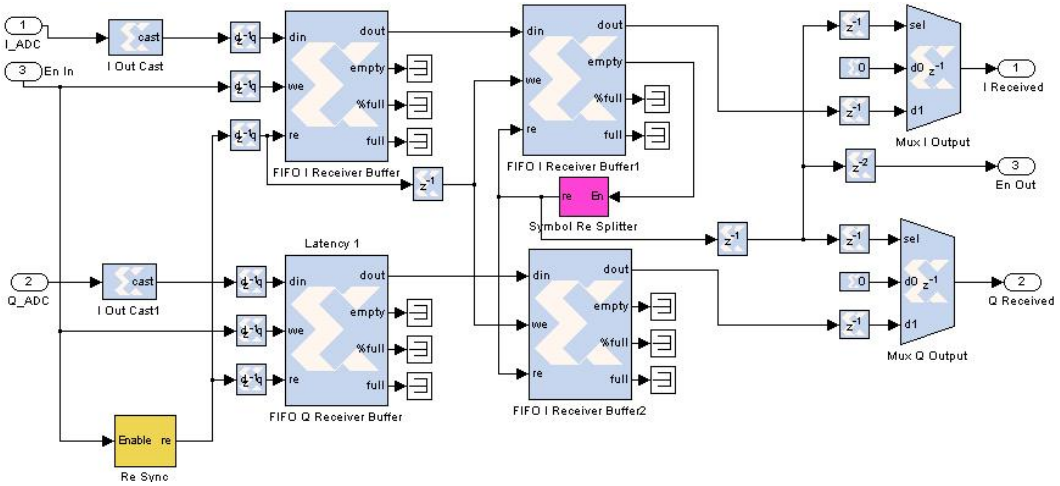


FRAME DETERMINATION BLOCK

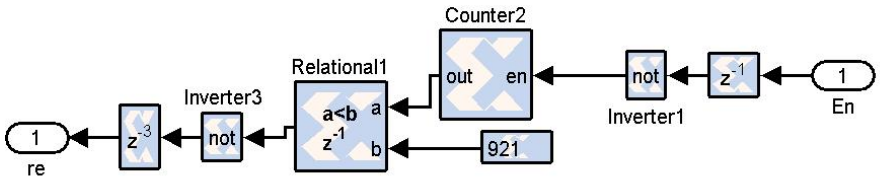


B.6. Input Buffer

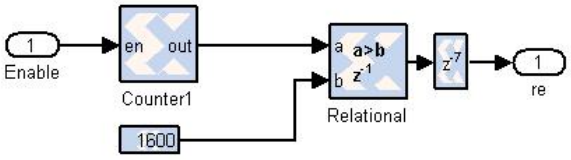
MAIN BLOCK



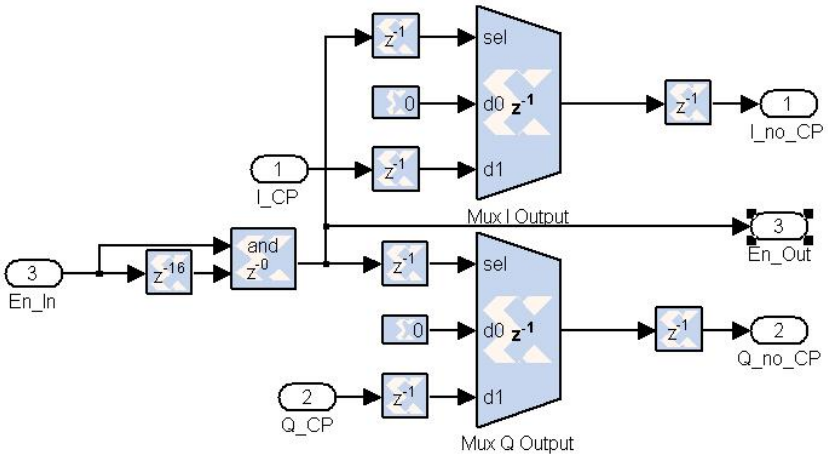
SYMBOL RESPLITTER BLOCK



RE SYNC BLOCK

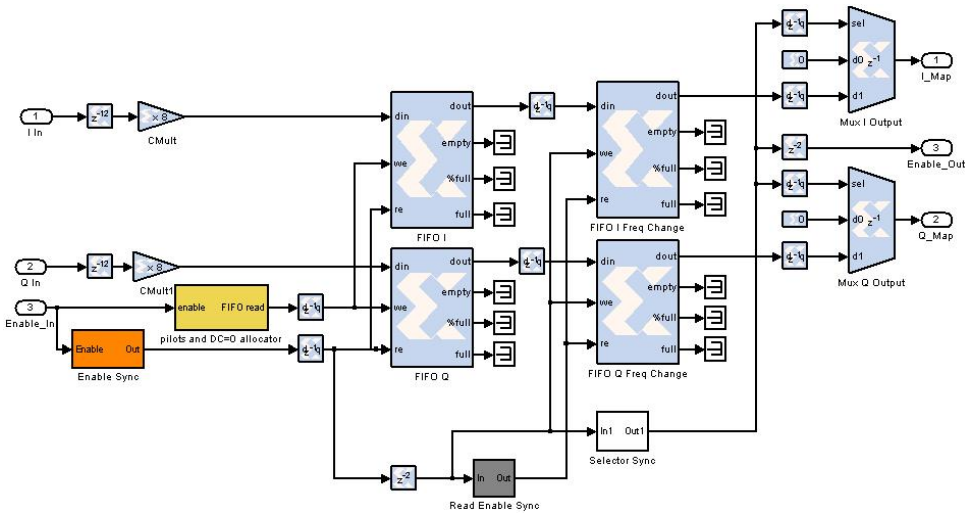


B.7. CP Removal

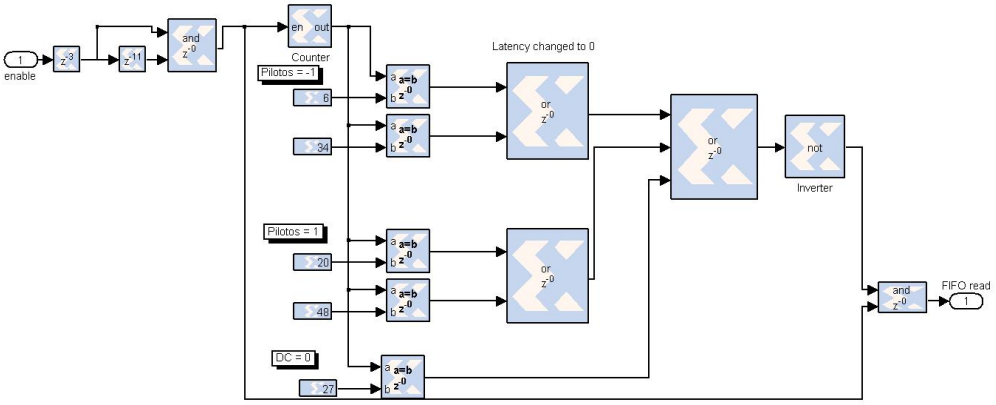


B.8. Pilot/DC Removal

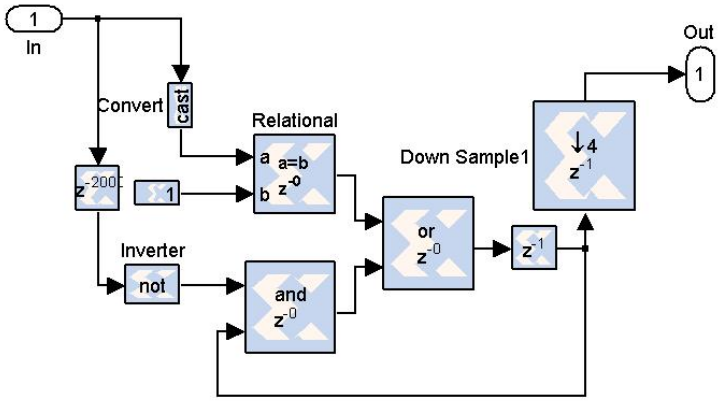
MAIN BLOCK



PILOTS AND DC=0 ALLOCATOR BLOCK

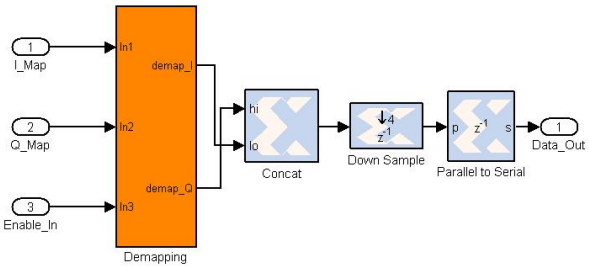


READ ENABLE SYNC BLOCK

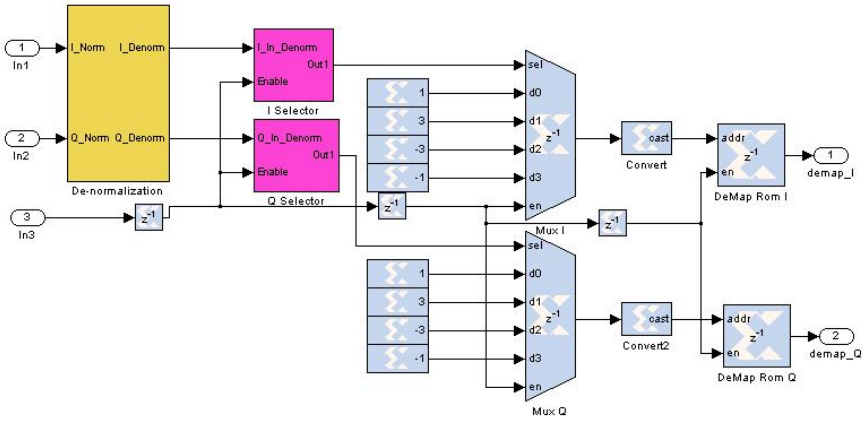


B.9 16-QAM Demodulator

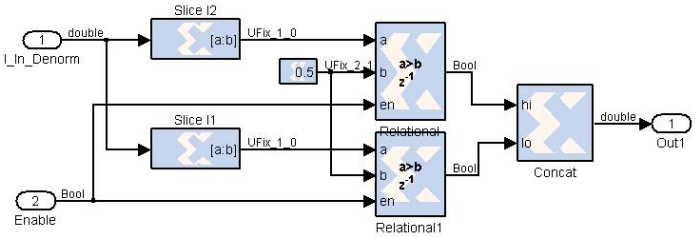
MAIN BLOCK



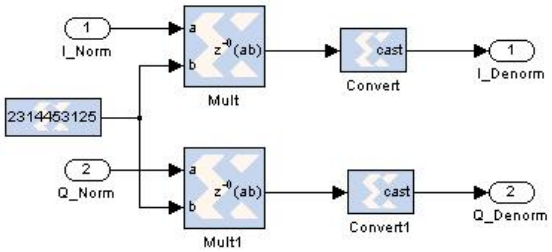
DEMAPPING BLOCK



IQ SELECTOR



DE-NORMALIZATION



Bibliography

[1] Nallatech, “XtremeDSP Development Kit-IV User Guide”, Issue 1, 2005.

[2] Analog Devices, “AD9772A Datasheet”, Revision C, 2008.

[3] Analog Devices, “AD6645 Datasheet”, Revision D, 2008.